COMPLETION TIME DRIVEN HYPER-HEURISTIC APPROACH FOR OPTIMISATION OF SCIENTIFIC WORKFLOW SCHEDULING IN CLOUD ENVIRONMENT

EHAB NABIEL MOHAMMAD

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

COMPLETION TIME DRIVEN HYPER-HEURISTIC APPROACH FOR OPTIMISATION OF SCIENTIFIC WORKFLOW SCHEDULING IN CLOUD ENVIRONMENT

EHAB NABIEL MOHAMMAD

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Ehab Nabiel Mohammad**

Registration/Matrix No.: **WHA100004**

Name of Degree: **DOCTOR OF PHILOSOPHY**

Title of Project Thesis ("this Work"): **Completion Time Driven Hyper-Heuristic Approach for Optimisation of Scientific Workflow Scheduling in Cloud Environment**

Field of Study: **Software Engineering (Computer Science)**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:
Designation:

COMPLETION TIME DRIVEN HYPER-HEURISTIC APPROACH FOR OPTIMISATION OF SCIENTIFIC WORKFLOW SCHEDULING IN CLOUD ENVIRONMENT

ABSTRACT

Effective management of Scientific Workflow Scheduling (SWFS) processes in a cloud environment remains a challenging task when dealing with large and complex Scientific Workflow Applications (SWFAs). The cost optimisation of SWFS approaches is affected by the inherent nature of SWFA as well as various types of scenarios that depend on the number of available virtual machines and size of SWFA datasets. However, current meta-heuristic based SWFS approaches lack the provision of satisfactory optimal solution, considering limited computational resources (e.g., virtual machines), longer execution time and high computational cost for a complex SWFA. Thus, the main objective of this research is to propose a Completion Time Driven Hyper-Heuristic (CTDHH) approach for cost optimisation of SWFS in a cloud environment. The first stage (i.e. formulation stage) of the research methodology involves an in-depth analysis of different cost optimisation perspectives of SWFS including aspects, parameters, challenges and approaches. The second stage (i.e. approach development stage) is the development of the proposed CTDHH approach, which includes two main parts, the cost optimisation model of SWFS and the dynamic hyper-heuristic algorithm. The proposed approach enhances the native random selection way of existing hyper-heuristic approaches by incorporating the best computed workflow completion time to pick a suitable algorithm from the pool of low-level heuristic algorithms after each run. The third and last stage (i.e. evaluation and analysis stage) aims at evaluating the proposed approach by considering two different experimental cloud environments: simulation-based environment and real-world based

environment. The performance of the proposed approach is evaluated by comparing it with four population-based approaches and an existing hyper-heuristic approach named Hyper-Heuristic Scheduling Algorithm (HHSA). Based on the results of the experiments, the proposed approach has proven to yield the most effective performance results for most of the considered experimental scenarios.

Keywords: Scientific workflow, workflow scheduling, cost optimisation, hyper-heuristic, cloud computing

PENDEKATAN HYPER-HEURISTIK DIDORONG OLEH MASA PENYELESAIAN UNTUK PENGOPTIMUMAN JADUALAN ALUR KERJA SAINTIFIK DALAM PERSEKITARAN AWAN

ABSTRAK

Pengurusan proses Aliran Kerja Penjadualan Saintifik (SWFS) yang berkesan dalam persekitaran awan masih merupakan suata cabaran apabila berurusan dengan Aplikasi Aliran Kerja Saintifik (SWFAs) yang besar dan kompleks. Kos pengoptimuman pendekatan SWFS dipengaruhi oleh sifat sedia ada SWFA serta pelbagai jenis senario yang bergantung kepada bilangan mesin maya yang tersedia dan saiz set data SWFA. Walau bagaimanapun, pendekatan SWFS berasaskan meta-heuristik yang sedia ada tidak mampu memperolehi penyelesaian optimum yang memuaskan, memandangkan sumber pengiraan terhad (seperti mesin maya), masa pelaksanaan yang lama dan kos pengiraan yang tinggi untuk SWFA yang kompleks. Oleh yang demikian, matlamat utama penyelidikan ini adalah untuk mencadangkan pendekatan Hyper-Heuristik didorong oleh Masa Penyelesaian (CTDHH) bagi pengoptimuman kos SWFS dalam persekitaran awan. Tahap pertama (iaitu peringkat penggubalan) metodologi penyelidikan melibatkan analisa mendalam mengenai perspektif pengoptimuman kos SWFS yang berbeza termasuk aspek, parameter, cabaran dan pendekatan. Peringkat kedua (iaitu peringkat pembangunan pendekatan) adalah pembangunan pendekatan CTDHH yang dicadangkan, yang mana ia merangkumi dua bahagian utama, model pengoptimuman kos SWFS dan algoritma hyper-heuristic dinamik. Pendekatan yang dicadangkan meningkatkan cara pemilihan rawak asli pendekatan hyper-heuristik yang sedia ada dengan memasukkan masa penyelesaian aliran kerja yang terbaik untuk memilih algoritma yang sesuai dari kumpulan algoritma heuristik peringkat rendah selepas setiap larian. Peringkat ketiga dan terakhir (iaitu peringkat penilaian dan analisis)

bertujuan untuk menilai pendekatan yang dicadangkan dengan mempertimbangkan dua persekitaran awan eksperimen yang berbeza: persekitaran yang berasaskan simulasi dan persekitaran yang berdasarkan dunia sebenar. Prestasi pendekatan yang dicadangkan itu telah dinilai dengan membandingkannya dengan empat pendekatan berasakan populasi dan satu pendekatan hyper-hueristik yang bernama Algoritma Penjadualan Hyper-Heuristik (HHSA). Berdasarkan hasil perbandingan eksperimen, pendekatan yang dicadangkan terbukti menghasilkan keputusan prestasi yang paling berkesan untuk semua senario eksperisen yang dipertimbangkan.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank the Almighty Allah for giving me this opportunity and for his spiritual help to complete this thesis. Second, I would like to thank my supportive supervisors, Professor Dr. Lee Sai Peck and Associate Professor Dr. Ling Teck Chaw for their ideal supervision and guidance without which this thesis would not have been completed. They sparked my interest in research and helped me to appreciate the value of looking beyond horizons of immediate practice.

Above all, I would also like to express my most sincere gratitude to my family for their spiritual support, and for being extremely thoughtful and encouraging during difficult times.

I would also like to thank my colleagues and friends (Saif Ur Rehman Khan, Sima Zamani, Chun Yong Chong and Abdullah Yousafzai) for their continuous encouragement throughout my study, without which I would not have been able to complete my thesis on time.

Furthermore, I would like to thank the staff of Faculty of Computer Science and Information Technology for their assistance. Last but not least, I would like to acknowledge the grant RG114-12ICT from the Ministry of Education, Malaysia.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

Workflow applications have been widely used by organisations to automatically process a set of tasks. Workflow automation brings along number of advantages to the employed organisation such as less errors during task processing, faster task execution than manual task management, and less cost in terms of execution time. In this research context, a workflow can be defined as series of jobs and these jobs represent one or several computational tasks based on their dependencies. These computational tasks can be any executable instances (e.g., load sets, report sets, programs, and data) with different structures (e.g., process, pipeline, data distribution, data aggregation, and data redistribution). In the literature, workflow applications have been classified into two main categories: (i) Business Workflow Application (BWFA), and (ii) Scientific Workflow Application (SWFA). The BWFA is defined as the automation of a business process in whole or part during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules (e.g., bank transactions and insurance claim applications) (Wieczorek, Hoheisel, & Prodan, 2009; de Oliveira, Ocaña, Baião, & Mattoso, 2012; Chandrakumar, 2013; X. Yang, Wallom, Waddington, Wang, & Blower, 2014; Poola, Garg, Buyya, Yang, & Ramamohanarao, 2014). Conversely, SWFA, also known as data and computational intensive scientific workflow application, mostly processes data flows together with the tasks' execution (Wieczorek et al., 2009; Ma, Gong, & Zou, 2009; Malawski, Figiela, Bubak, Deelman, & Nabrzyski, 2014; Tolosana-Calasanz, BañAres, Pham, & Rana, 2012; Kousalya, Balakrishnan, & Raj, 2017; Deldari, Naghibzadeh, & Abrishami, 2017; J. Lin, Luo, Li, Gao, & Liu, 2017; Park, Mei, Nguyen, Chen, & Zhang, 2017). A SWFA includes input scripts (e.g., scientific program or data), which can be used to produce, analyse and visualise the obtained results. Existing workflow applications

that fall in the category of SWFA include earthquake prediction application, biomedical application, and astrophysics application. SWFA has to deal with a large and complex set of tasks. For example, weather-forecasting system is one of the prime applications of SWFA that requires gathering of large-sized datasets (in the form of images) from satellites to efficiently analyse the captured data for accurate weather prediction. Furthermore, the SWFA can provide interactive tools to help scientists in better executing their own workflows and visualise the results in a real time manner. At the same time, it simplifies the process for scientists to reuse the same workflows and provides an easy-to-use environment to track and share output results virtually. However, SWFAs requires extra powerful processing resources to handle large and complex datasets and have been the subject of attention of researchers and scientists from the last two decades to provide more efficient solutions to handle such large and complex SWFAs.

One of the crucial challenges of SWFA is high data dependency which arises due to precedence constraints that require execution of preceding task (e.g., task A) before initiating dependent tasks (e.g., tasks B and C). In this case, the SWFA cannot start the execution of tasks B and C until task A has been accomplished. This is due to the reason that successful processing of considered tasks (B and C) relies on the resultant data as produced by the task A. At the same time, the SWFA contains a number of executable tasks, where each task may consume a large amount of data in order to be successfully executed using available computational resources. Thus, a Workflow Management System (WfMS) is required to effectively manage the submitted workflow tasks. A WfMS defines, manages, and executes workflows on available computing resources, where the workflow execution order is driven by a computer representation of the workflow logic. There are several stages that need to be performed by the WfMS in order to accomplish the submitted workflow tasks. Each stage of WfMS is responsible to process the submitted tasks based on different

underlying techniques. For instance, the modeling stage can be done using Directed Acyclic Graph (DAG) technique to highlight the precedence constraints of the submitted workflow tasks. In comparison to the modeling stage, scheduling (also referred as a global task scheduling) is considered as a core workflow processing stage of a WfMS. Scheduling is aimed to automatically assign and manage the execution of dependent tasks on shared resources through a scheduler. The purpose of scheduling the workflow tasks is to find the most suitable allocation of workflow tasks to available computational resources. The Scientific Workflow Scheduling (SWFS) mainly facilitates the computational processes between dependent workflow tasks and available computational resources. The SWFS process helps in mapping the dependent tasks to the available computational resources by following user requirements. However, scheduling is known as an NP-complete problem especially in the case of large and complex tasks, since there is no exact solution of the given workflow tasks (Z. Wu, Liu, Ni, Yuan, & Yang, 2013; Bittencourt & Madeira, 2011; Ramakrishnan, Chase, Gannon, Nurmi, & Wolski, 2011). Such type of NP-complete problem (i.e. scheduling) needs to be optimised to seek out an approximate solution with near polynomial time. Approximate solutions may exist using several methods as proposed by different researchers.

After processing the workflow tasks in modeling and scheduling stages, the WfMS needs to submit the scheduled tasks to the execution stage. The execution process of WfMS can be applied using various computational environments. For instance, researchers previously have used cluster computing, parallel computing and grid computing. After the emergence of cloud computing, researchers have started migrating their WfMS to the cloud computing environment as it offers more powerful, scalable, flexible, and virtualised features. However, the cloud computing environment demands more efficient SWFS approaches in order to handle several optimisation challenges including cost optimisation,

security, load balancing, reliability, and Quality of Service (QoS). Cost optimisation of SWFS could help in minimising the completion time and total computational cost. The cost optimisation challenge remains as an important consideration, due to its direct impact on both service consumers and service providers from different perspectives. The heuristic approaches have been widely used for the cost optimisation challenge of SWFS, and mainly used to efficiently determine the tasks' order and schedule them according to the best performance (Sakellariou, Zhao, Tsiakkouri, & Dikaiakos, 2007; Durillo, Prodan, & Fard, 2012). On the other hand, meta-heuristic methods are higher-level heuristic designed to select a heuristic to provide a sufficiently good solution to an optimisation problem like cost optimisation of SWFS. Meta-heuristic approaches (e.g., genetic algorithm) have also been effectively used to achieve optimised performance compared to other heuristic methods, but with some compromise on the computational time (Z. Wu et al., 2013; Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2012; Alkhanak, Lee, Rezaei, & Parizi, 2016; L. Liu, Zhang, Buyya, & Fan, 2017; Verma & Kaushal, 2017; Rodriguez & Buyya, 2017; Park et al., 2017). However, current state-of-the-art meta-heuristic based SWFS approaches lack the provision of satisfactory optimal solution within polynomial time, considering limited computational resources (e.g., virtual machines), longer execution time and high computational cost for a complex SWFA. Thus, there is a need to consider minimising both completion time (makespan) and total computational cost to provide a satisfactory cost-optimal solution. Cost optimisation performance of SWFS approaches is affected by the inherent nature of SWFA as well as various types of scenarios that depend on the number of available virtual machines and size of SWFA datasets. A large size of SWFA datasets can cause a significant increase in the dependency among workflow tasks, and this resultant tasks dependency ultimately increases the execution time required to process the SWFA. The cost optimisation performance of existing SWFS approaches is

still not satisfactory for all considered scenarios. Therefore, there is a need to propose a dynamic hyper-heuristic approach that can effectively optimise the cost of SWFS for all different scenarios. This can be done using hyper-heuristic approach by employing different meta-heuristic algorithms in order to utilise their strengths for each scenario.

In the literature, several approaches have been proposed for the cost optimisation challenge of SWFS by employing different types of approaches, e.g., heuristic, meta-heuristic. Yet, current state-of-the-art SWFS approaches lack in providing a dynamic cost optimisation solution for various types of scenarios that depend on the number of available virtual machines and size of SWFA dataset. Therefore, there is a need to propose a completion time driven hyper-heuristic approach for cost optimisation of SWFS by employing the strengths of existing meta-heuristics algorithms.

1.1 Research Background and Motivation

The Scientific Workflow Scheduling (SWFS) problem has recently attained more attention of researchers compared to the job scheduling. Job scheduling is a traditional way of scheduling that handles dependency among the submitted jobs. In contrast, SWFS aims at mapping and managing the execution of inter-dependent (i.e., precedence constraints) tasks on shared resources for applications with high complexity (N. Kaur, Aulakh, & Cheema, 2011). Therefore, SWFS approaches should be able to efficiently determine an optimised solution for large and complex SWFAs by considering precedence constraints between potential tasks. The SWFS problem is usually represented as a two-dimensional array, where the first dimension is the set of jobs/tasks and the second dimension is a set of available computational resources (e.g., VMs). Figure 1.1 illustrates an example of the two-dimensional representation of SWFS, where each task must be executed in one VM at the time. Also, it is important to consider the dependencies between the submitted tasks, which can directly affect the execution order of the tasks.

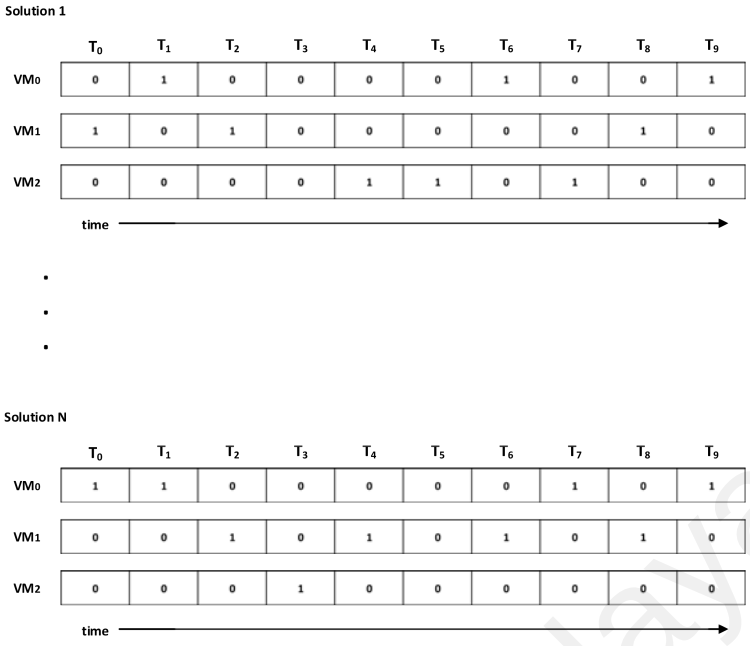

Figure 1.1: Two-dimensional representation of SWFS

The SWFS is required to manage the execution processes by facilitating the submitted workflow tasks and map these inter-dependent tasks to the available computational resources. In order to complete the SWFS processes, three main stages should be considered (Figure 1.2): (i) SWFA stage, (ii) WfMS stage, and (iii) computational environment stage. Each of the mentioned stages has to provide several specifications, such as user rules, format of the input/output data, task dependencies, functional and non-functional user requirements, to successfully accomplish the SWFS.

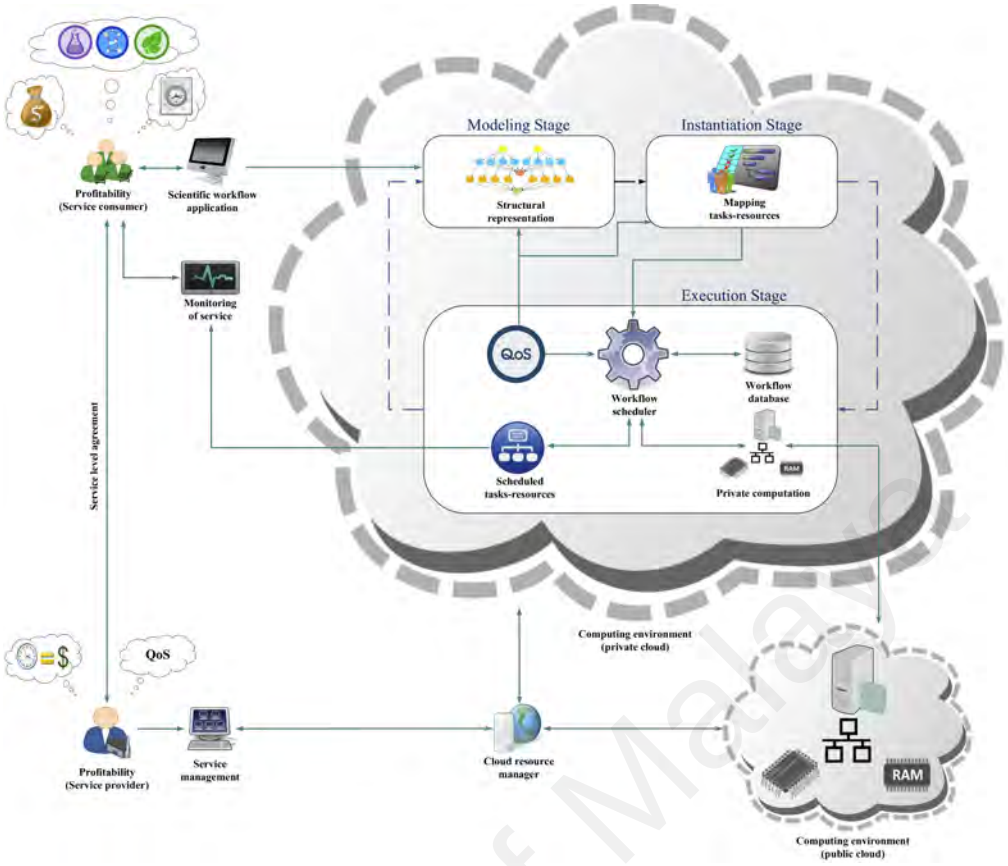

Figure 1.2: Process flow of scientific workflow scheduling

The ***Scientific Workflow Application (SWFA)*** is the first stage where the users (i.e. scientists) need to specify the nature of data. In other words, the SWFA receives user preferences about the execution order of tasks to be executed in the computational environment stage. The users' preference includes the precedence constraints of tasks, completion time and total computational cost. A number of inputs are required from users to successfully perform the SWFS. The main inputs are, inter-dependent tasks (e.g., programs) associated with their input data (e.g., images), along with the scripts, catalogues, and Application Program Interface (API) written using different programming languages (e.g., XML, Python, C, Perl, and Java) to represent the dependencies of the submitted workflow tasks. The expected outcome of SWFS (from the users' view) is the statistical and analysis data obtained from executing the workflow tasks. Thus, SWFA has several

advantages to the users. For instance, it simplifies the process for scientists to reuse the same workflows. Ultimately, SWFA enables the scientists with an easy-to-use environment to track and virtually share the obtained results.

The ***Workflow Management System (WfMS)*** stage is the second stage of SWFS processes. The information technology staff normally execute workflow tasks manually, which requires prior knowledge regarding resource availability and the estimated starting time for each workflow task (Dutta & VanderMeer, 2011; D. Yuan, Yang, Liu, & Chen, 2010; Miu & Missier, 2012; J. Lin et al., 2017; Park et al., 2017). However, manual task execution introduces many challenges including longer processing time, staff availability, impact on quality due to the staff's understandability, and high probability of failure occurrence. Thus, it is of vital importance to automate and optimise the SWFS process in order to achieve an efficient WfMS. Monitoring of tasks execution status (i.e. successful, failure, hold) is an important outcome of WfMS to the users. The WfMS can be implemented for different purposes including process management, process optimisation, system integration, achieving flexibility, and improving maintainability. The input of WfMS stage is a set of dependent tasks, available computational resources (heterogeneous or homogeneous resources), and the functional and non-functional requirements.

The first stage of scheduling the workflow tasks is the modeling stage (Figure 1.2), where the workflow application processes are redesigned based on workflow specifications containing the task definitions, tasks structural representation (e.g., Directed Acyclic Graph (DAG)), and user-defined QoS requirements. In contrast, the second stage is the instantiation stage, where the WfMS selects and reserves a suitable cloud model (i.e. private cloud, public cloud or hybrid cloud) based on the Service Level Agreement (SLA) to execute the workflow tasks as well as satisfy the defined QoS requirements. The SLA is a formal contract between the user (service consumer) and cloud service provider. The

scheduler (i.e. workflow engine) plays a crucial role in SWFS by allocating the given tasks to the available resources by considering their task dependencies. The scheduler coordinates the data, controls flows according to the workflow specifications obtained from the modeling stage, and employs the candidate computational resources that have been reserved in the instantiation stage. The output of this stage is a set of clustered jobs where each of these jobs represents as set dependent tasks. The clustered jobs are optimally planned to be executed in the resource computational environment (cloud computing) based on the SWFS algorithm (i.e. meta-heuristic).

The ***computational environment stage*** is responsible to execute the scheduled jobs. For SWFAs, the users need to employ more powerful and scalable computational resources to execute the large and complex SWFA tasks in an efficient manner. Cloud computing is an emerging technology of parallel and distributed computing, which heavily depends on the infrastructural support as provided by the grid computing. The SWFS in a cloud computing environment can bring several advantages. For instance:

- Resource sharing: cloud computing offers advanced services by sharing resources using the virtualisation notion with the help of internet technologies. Accordingly, sharing resources would support real-time allocation to fully utilise the available resources, while improving elasticity of cloud services. Thus, the SWFS in a cloud computing environment needs to consider the virtualisation infrastructure (e.g., virtual services and virtual machines) to efficiently facilitate the computational processes.
- Cost of resource usage: in view of the user's requirements (i.e. pay-as-you-go and on-demand services), cloud computing provides a flexible cost mechanism. In comparison to cloud computing, the other parallel and distributed computational environments (e.g., grid computing) follow a quota strategy to determine the

accumulated cost of requested services (Foster, Zhao, Raicu, & Lu, 2008). Hence, the other parallel and distributed computational environments have no flexible costing mechanism as in cloud computing.

The main components of the computational environment stage is the virtualisation technologies such as VMs, virtual server, virtual hosts, and virtual networks. These virtualisation technologies have been used to manage and share the underlying physical complements such as Central Processing Units (CPUs), memories, networks, and hard disks. Based on the WfMS requirements, VMs can be created on the top of the virtual server and hosts to provide a flexible sharing of the physical components with different software architectures (e.g., operating system) and hardware specifications (e.g., Random Access Memory (RAM) and CPU). Based on the WfMS requirements, each of these VMs can have different rules. For instance, in HTCondor that uses High Throughput Computing (HTC) to distribute the submitted tasks to available VMs, the Master VM has the main rules to manage and submit the jobs to VM Workers (slaves). In a cloud computing environment, VMs can have different formats based on the type of cloud computing models including private cloud, public cloud and hybrid cloud. In a private cloud model, the computational resources are configured, managed and owned by the users locally. Thus, the users have the full control of the computational resources. On the other hand, the public cloud model is used by the users to execute workflow tasks globally. To achieve this, there is a need from users to negotiate directly with the cloud service providers (e.g., Amazon, Google) based on the SLA. In comparison to private and public cloud models, the hybrid cloud model can provide users a more flexible computational environment to execute their workflow tasks based on their needs. For example, in a scenario where a private cloud is unable to execute submitted workflow tasks within the requested completion time and total computational cost, the WfMS can automatically utilise a public cloud to successfully

complete the execution of the submitted workflow tasks.

One of the most challenging problems of SWFS in cloud computing is optimising the cost of workflow execution (Saeid Abrishami, 2013; J. Yu & Buyya, 2006a; J. Li et al., 2015). The cost optimisation challenge of SWFS in cloud computing requires consideration of several aspects: inherent nature of SWFA as well as various types of scenarios that depend on the number of available virtual machines and size of SWFA datasets. However, considering all aspects of cost optimisation problem makes the SWFS process more complicated and requires a high amount of computational resources to meet the completion time (Rahman, Li, & Palit, 2011; Senna, Bittencourt, & Madeira, 2012; Grandinetti, Pisacane, & Sheikhalishahi, 2013; Ma et al., 2009; Alkhanak et al., 2016; L. Liu et al., 2017; Verma & Kaushal, 2017; Deldari et al., 2017; J. Lin et al., 2017; Park et al., 2017). In the literature, cost optimisation approaches of SWFS have been proposed aiming at minimising the total computational cost of SWFS (Z. Wu et al., 2013; Zheng & Sakellariou, 2013; Sahar Adabi, 2014; Szabo, Sheng, Kroeger, Zhang, & Yu, 2014). However, the state-of-the-art SWFS approaches require long completion time and total computational cost to execute the given SWFA tasks. A dynamic hyper-heuristic approach would provide a more optimal solution for the aforementioned problems (Alkhanak et al., 2016; L. Liu et al., 2017; Verma & Kaushal, 2017; Rodriguez & Buyya, 2017; Kousalya et al., 2017; P. I. Cowling & Chakhlevitch, 2007; Tsai, Huang, Chiang, Chiang, & Yang, 2014; Tsai, Song, & Chiang, n.d.).

1.2 Problem Statement

One of the most challenging processes of WfMS in a cloud computing environment is to schedule the submitted SWFA tasks to the available computational resources, while optimising the cost of executing the SWFA. The cost optimisation challenge of SWFS in cloud requires consideration of the following three main perspectives:

(i) There are strong inter-dependencies between the SWFA tasks. This computational-intensiveness of tasks introduces more complexity to the scheduling processes, since the data of the SWFA tasks needs to be transferred between the computational resources (i.e. VMs) of cloud computing.

(ii) There are different sizes of SWFA datasets that need to be considered by the scheduler, which significantly affect the execution time and execution cost. Thus, the data-intensiveness needs to be considered while calculating the completion time (makespan) and total cost of executing the tasks of SWFA on available resources (i.e. VMs).

(iii) There are different numbers of computational resources (VMs) based on the user requirements.

Accordingly, considering the abovementioned perspectives makes the SWFS process more complicated and requires a great amount of computational resources in terms of completion time and total computational cost.

Overall, the problem of this research can be stated as follows:

"Cost optimisation performance of SWFS approaches is affected by the inherent nature of the SWFA as well as various types of scenarios that depend on the number of available virtual machines and size of SWFA datasets. A large size of SWFA datasets can cause a significant increase in the dependency among workflow tasks, and this resultant tasks dependency ultimately increases the completion time required to process the SWFA. The cost optimisation performance of existing SWFS approaches is still not satisfactory for all considered scenarios. Therefore, there is a need to propose a dynamic hyper-heuristic approach that can effectively optimise the cost of SWFS for all different scenarios."

1.3 Research Objectives

A large size of SWFA datasets can cause a significant increase in the dependency among workflow tasks, and this resultant tasks dependency ultimately increases the completion time required to process the SWFA in available resources. Ultimately, any delay in completion time can negatively impact on cost optimisation of SWFS. Thus, the aim of this research is to propose a completion time driven hyper-heuristic approach for cost optimisation of SWFS in a cloud environment. Proposing such an approach can optimise the completion time as well as the total computational cost by dynamically selecting the most suitable meta-heuristic algorithm based on completion time performance of the employed meta-heuristic algorithms.

To address this aim, a number of Research Objectives (ROs) have been formulated.

RO1: *To analyse and investigate the main cost optimisation perspectives of SWFS in cloud environment.*

To achieve RO1, four Secondary Research Objectives (SROs) have been targeted:

SRO1.1: To identify the cost optimisation challenges of SWFS in the cloud environment.

SRO1.2: To identify the cost optimisation aspects of SWFS in the cloud environment.

SRO1.3: To identify the cost optimisation parameters of SWFS in the cloud environment.

SRO1.4: To identify the cost optimisation approaches of SWFS in the cloud environment.

RO2: *To develop a time driven hyper-heuristic approach for SWFS cost optimisation in cloud environment.*

To address RO2, two SROs have been targeted to determine as follows:

SRO2.1: To model the SWFS cost optimisation problem of the proposed approach.

SRO2.2: To propose a dynamic hyper meta-heuristic algorithm for cost optimisation of SWFS by employing several meta-heuristic algorithms.

RO3: *To evaluate and validate the performance parameters of cost optimisation of the proposed approach.*

To address RO3, the following four SROs have been targeted:

SRO3.1: To evaluate the proposed approach using two different cloud experimental environments, simulation-based and real-world based.

SRO3.2: To compare the proposed approach with the baseline approaches.

SRO3.3: To evaluate the proposed approach based on the considered scenarios (number of VMs, size of SWFA datasets).

SRO3.4: To analyse the data that are collected from the simulation-based and real-world based environments based on the completion time (makespan) and total computational cost parameters.

1.4 Research Questions

The research questions are formulated in order to closely adhering to the research objectives. In order to achieve the objectives of the research, the following Research Questions (RQs) have been formulated:

RQ1: *What are the key cost optimisation challenges of SWFS in the cloud environment?*

Answering RQ1 would help researchers to understand the reported cost optimisation challenges of SWFS in cloud computing. This helps in devising taxonomies, by identifying the relationship between existing cost optimisation challenges of SWFS in cloud environment. This comprehensive study would ultimately provide a complete description and

analysis of the cost optimisation challenges of SWFS in cloud computing. Furthermore, answering RQ1 would help in identifying the relevant cost constraints, which can play an important role in formulating the objective function and fitness function of the proposed approach.

RQ2: *What are the main cost optimisation aspects affecting the SWFS in the cloud environment?*

Answering RQ2 would help in identifying the important cost optimisation aspects of SWFS problem. This would also help in understanding the overall classification of the cost optimisation aspects to be considered in the proposed approach in this research.

RQ3: *What are the key cost parameters of SWFS in cloud computing, and how these parameters could affect the profitability of cost optimisation of SWFS?*

Answering RQ3 would help in identifying the relevant cost optimisation parameters, which are beneficial to formulate the cost model of the proposed approach.

RQ4: *What are the existing cost optimisation approaches for the SWFS problem?*

Answering RQ4 would help in identifying the relevant cost optimisation approaches for the SWFS problem. It also provides a clear understanding of strengths of the underlying optimisation, and limitations for all considered and reviewed approaches, which ultimately helps in selecting the most suitable approach for the identified problem of this research.

RQ5: *How to model the cost optimisation problem of SWFS?*

Answering RQ5 can help to understand the mapping and scheduling processes of workflow tasks by considering the scheduling stages along with completion time and total computational cost parameters.

RQ6: *How to propose a dynamic hyper-heuristic algorithm for cost optimisation challenge of SWFS?*

Answering RQ6 would help to propose a dynamic algorithm for cost optimisation of

SWFS that can achieve a satisfactory performance for all considered scenarios. This can be done by employing different meta-heuristic algorithms in order to utilise their strengths for each scenario.

RQ7: *What are the key experimental cloud environments that need to be considered for the evaluation of the proposed approach?*

Answering RQ7 can help in choosing the most effective and accurate experimental environment tools to evaluate the proposed approach in cloud computing.

RQ8: *What are the most relevant baseline and existing hyper-heuristic approaches that need to be considered to evaluate the proposed approach?*

Answering RQ8 would help in determining the most relevant baseline and existing hyper-heuristic approaches to be compared with the proposed approach in an efficient way.

RQ9: *How to evaluate the computational-intensiveness and data-intensiveness of the proposed approach?*

Answering RQ9 can help to evaluate the performance of the proposed approach by considering different scenarios based on the size of SWFA datasets and the number of VMs.

RQ10: *Would the proposed approach lead to results that are better than the considered baseline and existing hyper-heuristic approaches?*

Based on the completion time and total computational cost results, answering RQ10 would help to improve the behavior and performance of the proposed approach, compared with the baseline and existing hyper-heuristic approaches.

The research questions are formulated in order to closely adhering to the research objectives. Table 1.1 shows the relationships between the research objectives, the research questions, and research methods.

Table 1.1: Relationships between the research objectives, the research questions, and research methods

Resaerch Questions	Research Objectives	Research Methdology
RQ1	*SRO1.1*	*Literature Review*
RQ2	*SRO1.2*	
RQ3	*SRO1.3*	
RQ4	*SRO1.4*	
RQ5	*SRO2.1*	*Proposed Approach*
RQ6	*SRO2.2*	
RQ7	*SRO3.1*	*Evaluation and Analysis*
RQ8	*SRO3.2*	
RQ9	*SRO3.3*	
RQ10	*SRO3.4*	

1.5 Proposed Solution

In this research, a dynamic hyper-heuristic solution is proposed by employing four meta-heuristic algorithms for the cost optimisation problem of SWFS in a cloud computing environment. The existing population-based meta-heuristics solutions have shown good performance for the optimisation of the large search space problem. In contrast, single-based meta-heuristic solutions do not exhaustively search within the scheduling problem space, yet they use different underlying strategies to find the desired solution based on defined fitness criteria. Therefore, the population-based meta-heuristic solution takes less computational effort as compared to single-based solution while it can often find good solutions. However, each of these approaches has their strengths and limitations, which affect the SWFS processes.

The hybrid meta-heuristics uses the best features of two or more traditional meta-heuristics (e.g., Genetic Algorithm, Ant Colony Optimiation) in each iteration to provide a better optimal solution. Due to the complexity of hybrid meta-heurtstics method, it might take a longer convergence time than the traditional meta-heuristics for each iteration. To address this limitation, a completion time driven hyper-heuristic approach has been proposed in this research. Ultimately, it helps in optimising the completion time and total

computational cost of SWFS in cloud environment. The proposed approach contains two main parts, the cost optimisation model of SWFS and the dynamic hyper-heuristic algorithm. The cost optimisation model of SWFS can help to understand the mapping and scheduling processes of workflow tasks by considering the scheduling stages along with completion time and total computational cost parameters. While the proposed hyper-heuristic algorithm is considered as a new advanced technique that is capable of accelerating the run-time of a meta-heuristic algorithm. Hyper-heuristic solution is an emerging class of meta-heuristic search-based approaches that are combined in such a manner that allows utilising the maximum strengths of employed meta-heuristic to obtain an optimal solution. There are only few works that have considered utilising hyper-heuristic for SWFS, while hyper-heuristic can always find the most cost optimal solutions for different scenarios. The proposed algorithm employs four well-known population-based meta-heuristic algorithms, which act as Low Level Heuristic (LLH) algorithms (i.e., genetic algorithm, particle swarm optimisation, invasive weed optimisation, and hybrid invasive weed optimisation). In addition, the proposed algorithm enhances the native random selection way of existing hyper-heuristic solutions by incorporating the best computed workflow completion time to act as a high-level selector to pick a suitable algorithm from the pool of LLH algorithms after each run. The main aim of the proposed approach is to reduce the completion time and total computational cost to execute the SWFA. Based on the lowest achieved completion time, the proposed algorithm dynamically guides the searching processes to find an optimal solution by continuously sorting the computed time scores (i.e. completion times of previous runs) of all the employed LLH algorithms for each considered scenario and after every run. The computed time scores are listed in a scoreboard table. Next, for each single run, the high-level selector adopts the LLH algorithm that has the lowest computed time score for each scenario. The proposed dynamic hyper-heuristic

algorithm continuously updates the scoreboard table by replacing the existing time score with the lowest computed time score, which ultimately affects the total computational cost value for that run. Finally, based on the scoreboard table, the proposed approach selects the most appropriate LLH algorithm for the next run. Consequently, the mechanism of the proposed completion time driven hyper-heuristic approach becomes more effective in allowing to reuse and utilise the maximum strengths of the employed LLH algorithms in searching for the optimal solution of the targeted cost optimisation problem.

1.6 Scope of the Research

According to Software Engineering Body of Knowledge (SWEBOK) (Bourque & Fairley, 2014; Abran & Bourque, 2004) and international standard ISO/IEC TR 19759:2005 (Bourque & Fairley, 2014), there are five main stages in the development life-cycle of any standard software, which include requirements engineering, software design, software construction, software testing, and software maintenance. As discussed earlier, the main aim of this research is to propose a completion time driven hyper meta-heuristic search-based approach for cost-optimisation of SWFS in a cloud environment. Meta-heuristic is a sub-branch from Software Design Strategies and Methods (SDSM) (Harman, Lakhotia, Singer, White, & Yoo, 2013). The search-based method is one of the three main classes of SDSM heuristic methods, formal methods, and prototyping methods. In order to solve the cost optimiation problem of SWFS, it is required to adopt a hyper-heuristic approach, since it can find different alternative (approximate) solutions in a polynomial time. The search-based software engineering is one of the most popular methods that uses various optimisation techniques to address optimisation problems in software engineering (Harman et al., 2013). Figure 1.3 shows the scope of this research based on the SWEBOK guidelines (Bourque & Fairley, 2014; Abran & Bourque, 2004), where the highlighted boxes specify the general focus of this research.

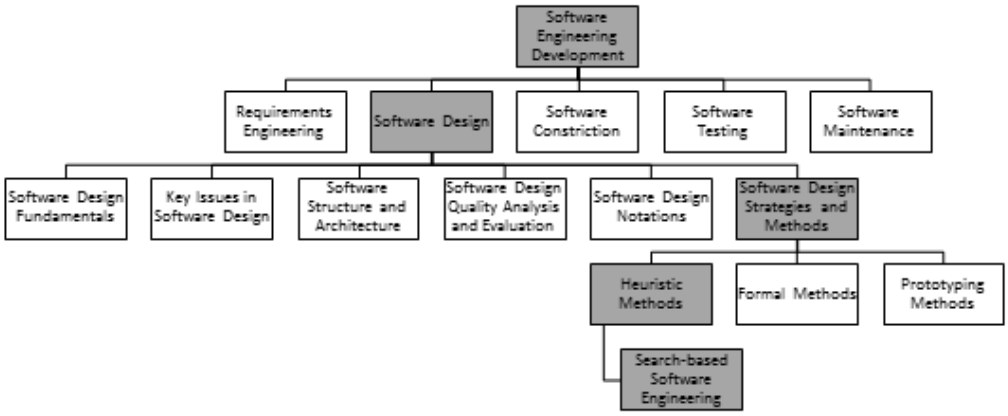

Figure 1.3: The scope of this research

1.7 Significance of Research

The outcomes of this research will help academic researchers and practitioners working in the area of SWFS in cloud computing environment. As previously mentioned, the main aim of the research is:

"To propose a completion time driven hyper-heuristic approach for cost optimisation of SWFS in a cloud environment"

Based on this aim, the following are main expected outcomes of this research:

- A completion time driven hyper-heuristic approach for cost optimisation of SWFS in a cloud environment. The proposed approach helps in optimising the completion time (makespan) and total computational cost of SWFS in cloud computing for all considered scenarios. This will ultimately reduce the cost for service consumers (i.e. scientists). At the same time, this reduced cost will increase the profitability for service providers (i.e. cloud service providers) towards utilising all computational resources to achieve a competitive advantage over other cloud service providers.

- The proposed approach has a direct impact on satisfying the functional QoS requirements by reducing the response time for service consumers. Furthermore,

it is expected that the proposed approach helps in saving energy and time of the service providers, by wisely utilising the computational resources (VMs).

- The proposed approach provides an efficient platform to optimally schedule workflow tasks by handling data-intensiveness and computational-intensiveness of SWFA.

- Applications developed based on the proposed approach can result in cost and time saving. This can improve the cloud computing services by providing more economic services to cloud users by satisfying the functional QoS user requirement for a number of potential SWFAs.

- Several benefits can be achieved from conducting the extensive literature review. The outcome of this research would be helpful for the academic researchers in providing clearer and complete understanding of the cost optimiation topic of SWFS in cloud computing, by providing the following expected outcomes:

* Several taxonomies of cost optimiation for SWFS challenges, aspects, parameters, and approaches.

* Correlation between the cost parameters and their profitability to service consumers and services providers.

* Future opportunities in this field of research. This would open new doors for high impact research that engenders innovative values through SWFS and cloud computing.

1.8 Thesis Organisation

This section outlines the thesis organisation that provides an overview of the chapters based on the defined research objectives of this work. Figure 1.4 illustrates the relationship

between the thesis chapters as well as the correlation between the thesis chapters and their associated research objectives and research questions. The following is a brief explanation of each of the thesis chapters:

Chapter 1: Introduction This chapter provides a background information to the research problem statement, research objectives and questions, proposed solution, scope of the research, and significance of the research.

Chapter 2: Literature review This chapter discusses the conducted literature review that supports this study. The literature review has helped in determining the accepted cost optimisation criteria while developing the cost optimisation approach of SWFS. Furthermore, based on the literature review, a completion time driven hyper-heuristic approach is proposed for cost optimisation of SWFS in cloud computing.

Chapter 3: Research methodology This chapter provides an overview on the main stages and activities that have been adopted in this research.

Chapter 4: Proposed approach This chapter discusses the proposed cost optimisation model of SWFS as well as the proposed dynamic hyper-heuristic algorithm for cost optimisation challenge of SFWS in cloud environment.

Chapter 5: Evaluation and analysis using simulation environment This chapter provides significant details about experimental setup of the considered evaluation using simulation environment. The analysis of the collected data from the simulation environment is explicitly discussed in this chapter along with the obtained results.

Chapter 6: Evaluation and analysis using real-world environment This chapter provides a comprehensive explanation about the real-world experimentation environment that has been used as another way to evaluate the proposed approach. Similar to Chapter 5, the analysis of the collected data from the real-world environment has been performed along with the discussion of the extracted results.

Chapter 7: Conclusion and future trends This chapter provides a summary of the highlighted research objectives, limitations of research and future work that can be expanded from this study.

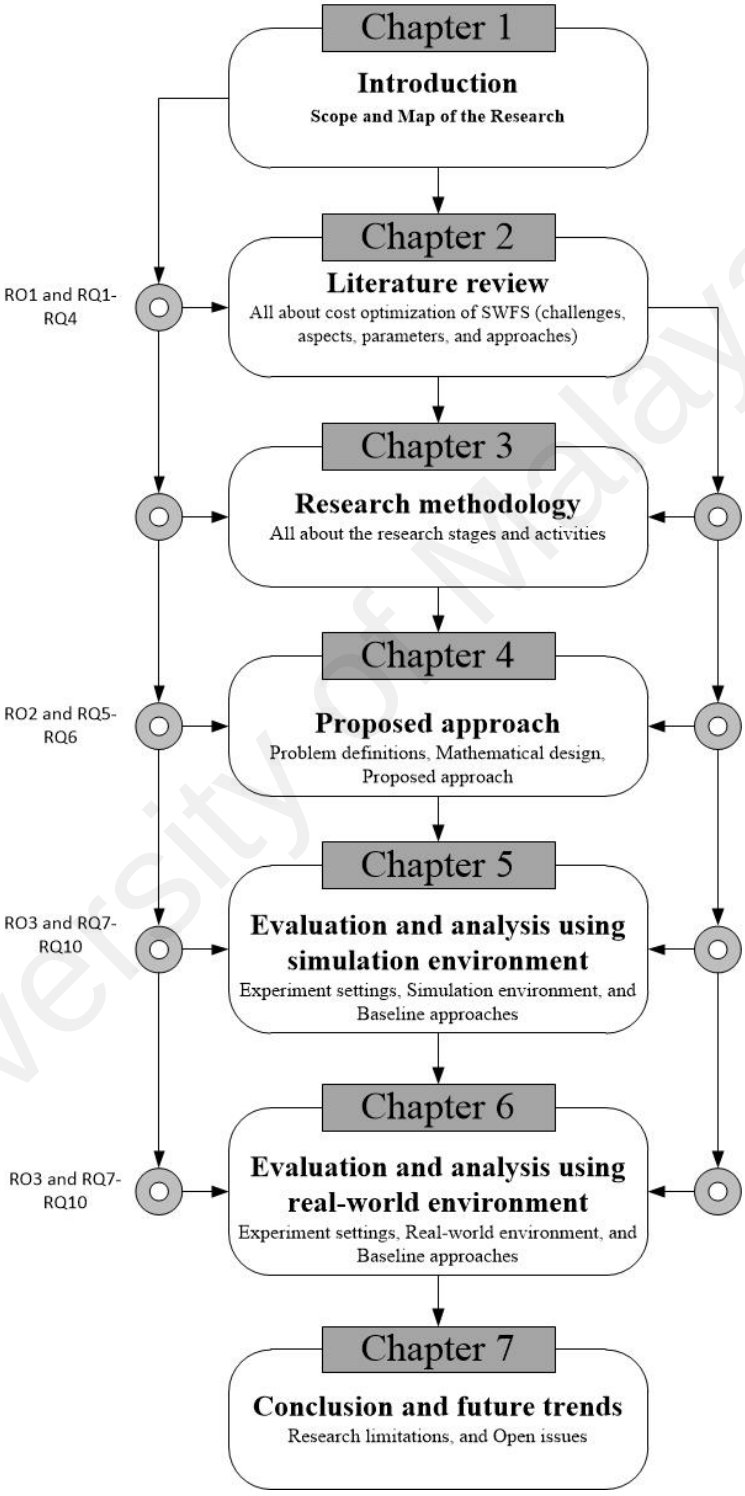

Figure 1.4: Thesis organisation

CHAPTER 2: LITERATURE REVIEW

The cost optimisation of Scientific Workflow Scheduling (SWFS) especially in cloud computing remains an important challenge for both service consumers and service providers. The current work analyses the cost optimisation problem for SWFS in cloud computing. After careful selection of the relevant works in this field of study, the cost optimisation aspects, parameters, challenges and approaches of SWFS have been identified, while scheduling workflows in cloud computing. This chapter introduces three main classifications (i.e. aspects, parameters and challenges) of cost optimisation SWFS, which is one of the major contributions of this thesis. The proposed classifications aim at providing a solid foundation for developing the proposed SWFS approach in this study to meet the demands of future SWFAs. This study has classified the related works according to the devised classifications and identified a correlation between cost optimisation parameters and profitability of SWFS. Besides, the chapter presents the relevant cost and time formulas, which are used to determine the cost optimisation model of SWFS by considering a number of relevant time and cost parameters. Moreover, the chapter provides several recommendations for developing a completion time driven hyper-heuristic approach for cost optimisation of SWFS in cloud environment.

2.1 Cost Optimisation Aspects of SWFS

Several aspects need to be considered while scheduling the SWFA tasks. This section presents a classification for aspects of cost optimisation SWFS approaches in cloud computing. Figure 2.1 presents our classification of cost optimisation aspects of SWFS based on eight main classes: computing environment, optimisation method, structural representation, profitability, scheduling technique, workload type, optimisation criteria, and QoS constraints.

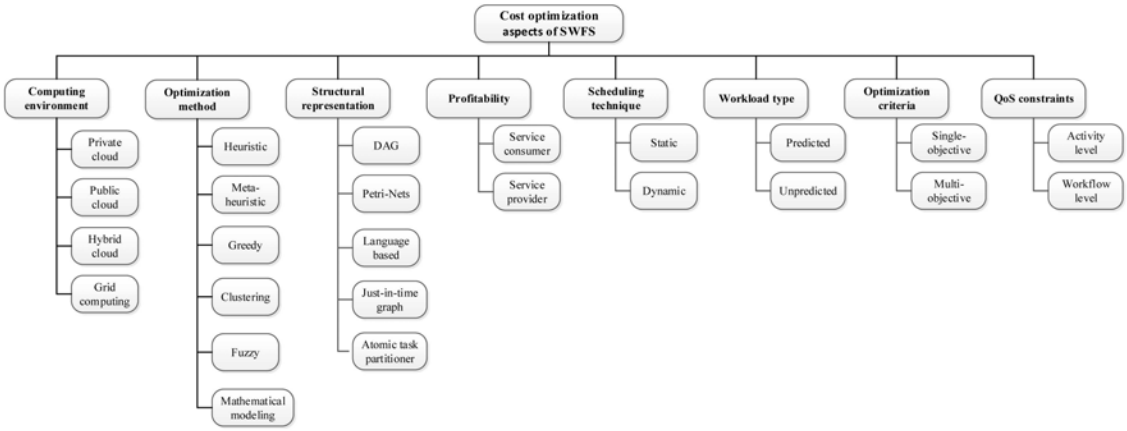

Figure 2.1: A classification of cost optimisation aspects of SWFS

In the following sub-sections, each aspect of the presented classification has been briefly discussed.

2.1.1 Computing Environment

The cost of executing a SWFA is usually affected by the computational environment due to the fact that it has direct impact on resource utilisation. For example, scientists should select the computing environment based on their requirement which could be related to the size of application, privacy of the used data and other QoS constraints (e.g. budget, and deadline). Therefore, each computing environment has a different specification, which ultimately affects the total computional cost of SWFS. In this section, three main computing environments have been considered. The computing environments are: (i) private cloud, (ii) public cloud, and (ii) hybrid cloud.

Private cloud: due to data privacy and limited budget constraints, most of the service consumers select a private cloud. The operational cost is usually not taken into consideration and the resources' usage cost is also not measured (Bittencourt & Madeira, 2011; Senna et al., 2012). The total computional cost of SWFS in the private cloud model can normally be calculated by adding computation cost to communication cost.

Public cloud: this computing type is usually selected when service consumers need to execute a large SWFA, which can not be executed locally. The total computional cost of SWFS includes the cost per time unit of using cloud resources (Ostermann, Prodan, & Fahringer, 2010). The communication cost value of executing workflow in the public cloud is normally assumed to be zero, because of the assumption that all resources are built into the same computational infrastructure.

Hybrid cloud: this type of computing can give a flexible specification for the required resources by service consumers. The scheduler is required to consider the load balancing of submitted work in order to fully utilise the private and public clouds. However, this model adds more complexity to the cost of the consumed services. Hence, the cost of SWFS can be calculated by adding the SWFS cost in a private cloud to the SWFS cost in a public cloud. Another hybrid cloud scenario is where the total computional cost of SWFS is defined as data transfers from and to the cloud (Ramakrishnan et al., 2011; Bittencourt, Madeira, & Da Fonseca, 2012). The available bandwidth into the connected processing resources of the hybrid cloud affects the makespan (Malawski, Juve, Deelman, & Nabrzyski, 2012; X. Liu et al., 2011, 2010; W.-n. Chen, Shi, & Zhang, 2009). Therefore, the bandwidth cost for the hybrid cloud environment can be defined as the cost that the service provider charges to service consumers per the amount of data transferred ($/GB).

Grid computing: this type of computing provides an optimal solution that can meet the user's requirements by providing scalable and flexible solutions for considered applications (Z. Wu et al., 2013). The cloud based task scheduling differs from the grid based scheduling in the following two ways:

- Resource sharing: cloud computing offers advanced services by sharing resources using the virtualization notion with the help of internet technologies. Consequently, it supports real-time allocation to fully utilize the available resources while improving

elasticity of cloud services. Thus, the scheduler in a cloud workflow system needs to consider the virtualization infrastructure (e.g., virtual services and virtual machines) to efficiently facilitate the computational processes. In contrast, grid computing allows allocating a large cluster of resources in a shared mode. Therefore, it supports batch processing and resources will be available once they are released by other users.

- Cost of resource usage: cloud computing provides a flexible costing mechanism considering the user's requirements (i.e. pay-as-you-go and on-demand services). On the other hand, grid computing follows a quota strategy to determine the accumulated cost of requested services (Foster et al., 2008). Therefore, grid computing has no flexible costing mechanism as in cloud computing.

2.1.2 Optimisation Method

Optimisation method is considered one of the most important cost optimisation aspects due to its direct impact on task-resource mapping processes. Several methods (i.e., rule-based, search-based, coverage-based) have been proposed in the literature to find an optimal solution for the total computational cost of executing the SWFS in cloud computing environment. Heuristic methods have been widely used for the scheduling problem. The heuristic methods efficiently determine the tasks' order and schedule them according to the best performance (in terms of effectiveness and accuracy) (Sakellariou et al., 2007; Durillo et al., 2012). On the other hand, meta-heuristic methods (e.g. genetic algorithm) have also been effectively used to achieve improved performance compared to other heuristic methods, but with some compromise on the execution time (Z. Wu et al., 2013; Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2012).

2.1.3 Structural Representation

Due to the complex nature of SWFAs, structural representation is the first stage of any Workflow Management System (WfMS) that is used to simplify the submitted scientific activities to the scheduler. Several types of structural representation methods have been adopted in the literature to represent the tasks' dependency (precedence constraints) of SWFS: (i) Graph-based modeling methods: DAG (Stephanakis, Chochliouros, Caridakis, & Kollias, 2013; Y. Yuan, Li, Wang, & Zhu, 2009, 2007; J. Li, Su, Cheng, Huang, & Zhang, 2011; Y. Tao, Jin, & Shi, 2007; Arabnejad & Barbosa, 2014), Petri Nets (Van Hee, 2004; Y. Liu, Zhu, Chen, Li, & Deng, 2014), and (ii) Language-based modeling tools (i.e., XML Process Definition Language (XPDL)) (Kwok & Ahmad, 1999). For each type of these methods, there are cost parameter representations (e.g. computation cost, and communication cost). For instance, the DAG based method is the most popular method used in the state-of-the-art approaches to estimate the execution cost of different available resources for every task, which represents the overall computational cost. In addition, the time to communicate data between resources is given, which represents communication cost (e.g., bytes to transmit).

2.1.4 Profitability

As shown in Figure 2.1, due to the importance of the cost optimisation of SWFS in cloud computing for different WfMS users, in order to have a deeper understanding on the proposed cost optimisation approaches, the reviewed state-of-the-art cost optimisation of SWFS approaches have been classified into two groups with respect to profitability: (1) approaches whose main goal is the service consumers' profitability, and (2) approaches whose primary goal is the service providers' profitability (Z. Wu et al., 2013; Salehi & Buyya, 2010; W.-N. Chen & Zhang, 2009; Y. Yuan, Li, & Wang, 2006; Malawski et al., 2012; Jiang, Huang, Chang, Gu, & Shih, 2011). The service consumer represents a person

or organisation (which could be a scientist or researcher) that uses the cloud computing services (i.e. Infrastructure as a Service (IaaS), Software as a Service (SaaS) or Platform as a Service (PaaS)) in order to execute the scientific application (Mell & Grance, 2009). Conversely, the service provider represents a company or organisation (which could be any cloud service provider) that offers cloud computing services to service consumers (persons or organisations) with different QoS constraints and prices (Z. Wu et al., 2013; Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013).

When a cloud service consumer requests a service from the service provider, a vital matter of reducing the leasing cost (for the service consumer) arises, while the contribution to lower the overall execution cost of workflow (for the service provider) increases (Ostermann et al., 2010; Szabo & Kroeger, 2012; Bhise & Mali, 2013; Szabo et al., 2014). In contrast, for the service provider, the aim is to reduce the cost of leasing time of the cloud resources (Bittencourt & Madeira, 2011; Pandey, Wu, Guru, & Buyya, 2010; K. Liu et al., 2010; Y. Yang et al., 2008; Ostermann et al., 2010; W.-N. Chen & Zhang, 2009; Saeid Abrishami, 2012). Therefore, time optimisation is profitable for the service provider by reducing the cost of maintaining resources. Consequently, this gives an advantage to cloud service consumers since it will reduce the cost of workflow execution (S. Kaur & Verma, 2012).

2.1.5 Scheduling Technique

The scheduling technique represents the mechanism that the scheduler chooses to schedule SWFA tasks that is strongly related to the cost of the utilised resources. Two types of techniques have been adopted in state-of-the-art cost optimisation of SWFS approaches: (i) static technique, and (ii) dynamic technique.

Static technique: Static technique requires the scheduler to know in advance the characteristics of all the scientific tasks, including their sizes, service demands and

estimated execution cost. Also, the static techniques are performed under two assumptions: (i) the tasks arrive simultaneously to the resources, and (ii) the resource's available time is updated after each task is scheduled (Nargunam & Shajin, 2012; Arabnejad & Barbosa, 2014). The static technique can efficiently schedule large workflows on large data centers. Another advantage is that it is easier to adapt a static technique based on the scheduler's perspective. Furthermore, it is more user friendly as the precomputed schedule allows quoting a price for the computation. In addition, it allows the service consumers to choose from multiple scheduling options according to the price and time constraints (Dutta & VanderMeer, 2011; Deng, Kong, Song, Ren, & Yuan, 2011).

Dynamic technique: Dynamic technique is more flexible than static scalability where scientific tasks are dynamically available (continuous stochastic stream) for scheduling. However, it is more complex than the static scheduling technique since it needs to update the system information on the fly (Nargunam & Shajin, 2012). The main advantage of dynamic strategy is that it can be adopted when a task set or a resource set is heterogeneous. For instance, not all tasks arrive simultaneously, or some resources are off-line at intervals (Fida, 2008). The other advantage is that it considers only few required parameters in advance. Due to the aforementioned advantages, the dynamic scalability is more suitable for executing the on-demand workflow applications in cloud environments.

2.1.6 Workload Type

Two types of workload methods, which are predicted mode and unpredicted mode, have been adopted in SWFS approaches based on their method of loading the tasks to the scheduler in cloud computing. The workload type can affect estimation.

Predicted (batch mode): in predicted or batch mode (also referred as latter mode), the tasks are first collected as a group of problems that are examined for scheduling at prescheduled times (predefined moments). Thus, it is better to map the tasks for suitable

resources depending on their characteristics (Nargunam & Shajin, 2012). This enables the predicted mode to determine about the actual execution time for a larger number of tasks (Dong, 2009). One of the main advantages of this mode is to maximise the throughput while minimising the turnaround time (the time between task submission and its completion) (Zheng, 2010).

Unpredicted (on-line mode): unpredicted or on-line mode (also referred as former mode) where tasks are scheduled to a resource as soon as they arrive for execution and there is no waiting for the next time interval on available resources at that moment (Nargunam & Shajin, 2012; Dong, 2009). In this mode, each task is scheduled only once and hence the scheduling result cannot be changed. Therefore, unpredicted mode is suitable for the scheduling scenarios where the arrival rate is low (X. Lin & Wu, 2013; Dong, 2009; Zheng, 2010).

2.1.7 Optimisation Criteria

In order to propose an optimal solution for the SWFS problem, the total computional cost of executing workflow tasks needs to be minimised. The reviewed SWFS approaches can be classified into two main classes: (i) Single-objective optimisation based approaches, and (ii) Multi-objective optimisation based approaches. In the literature, researchers have mainly focused on optimising cost parameters only. Thus, the approaches which only consider execution cost or time (but not both) are referred as *Single-objective optimisation* based approaches, since they only target at optimising the cost of SWFS problem. Some approaches have focused on minimising the execution cost (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013; H. Liu, Xu, & Miao, 2011; Genez, Bittencourt, & Madeira, 2012; Saeid Abrishami, 2012; Zeng & Wang, 2013; L. Zhao, Ren, Li, & Sakurai, 2012), while other approaches have focused on the execution time (Nargunam & Shajin, 2012; Ramakrishnan et al., 2011; Stevens et al., 2009; Tanaka & Tatebe, 2012; Chunlin & Layuan,

2006). However, due to the rapid development of the provided computing services (e.g., pay-as-you-go, and on-demand), many other constraints (e.g. QoS constraints) should also be considered to optimise the cost of SWFS. Due to the above-mentioned reasons, the complexity of approaches has increased to a great extent, which ultimately demands to handle the trade-offs between the cost and other affected constraints. Nevertheless, a group of services may have the same requirements so they can complete similar tasks or activities with different execution cost, execution time, availability and reliability (Y. Yuan et al., 2006). These approaches are referred as *Multi-objective optimisation* based approaches. Therefore, majority of the recent approaches have adopted hybrid and hyper techniques for the heuristics and meta-heuristic methods to obtain an optimal solution.

2.1.8 QoS Constraints

QoS has a major impact on SWFS in cloud computing, since the success of computational tasks heavily depends on the desired QoS levels (Lingfang, Veeravalli, & Xiaorong, 2012; Q. Wu et al., 2013; Saeid Abrishami, 2012; Topcuoglu, Hariri, & Wu, 2002). For multi-objective problems, such as SWFS, there are several QoS constraints that must be taken into consideration for a given service when designing an efficient WfMS in cloud computing (Bittencourt & Madeira, 2011; Delavar & Aryan, 2012; Sakellariou & Zhao, 2004a; S. Zhang, Liu, Wang, & Zhang, 2013). The service providers must consider satisfying the service consumers' QoS requirements based on Service Level Agreement (SLA). Therefore, for a scheduling process, the QoS has a direct effect on each stage of a typical workflow instance.

In this section, the QoS constraints for each of the reviewed cost optimisation approaches in Table 2.1 (as multi-objective optimisation criteria) have been identified. Furthermore, we have considered another important QoS aspect, which is the way of handling the QoS constraints in SWFS approaches. There are two methods to consider QoS for cost

optimisation of SWFS approaches in cloud computing as shown in Figure 2.2.

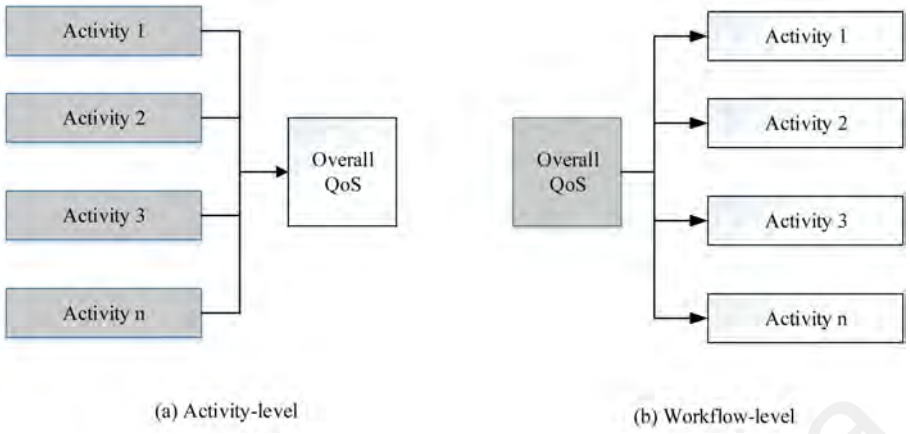

Figure 2.2: Methods for considering QoS constraints

The first method is to allow the users to assign activity-level QoS constraints, and then the overall QoS can be assessed by computing the QoS constraints of all individual activities based on the specific QoS model. For example, a workflow reduction algorithm can be employed to calculate the deadline for the entire workflow based on the desired execution time of individual workflow activities (Saeid Abrishami, 2012; X. Liu, 2012).

The second method is to assign QoS constraints at workflow-level where users need to define the overall workflow QoS requirements, and then the workflow system uses automatic strategies to assign local and activity-level QoS constraints to the workflow segments and individual activities. For example, a deadline assignment approach such as Equal Slack and Equal Flexibility (Xu, Cui, Wang, & Bi, 2009)) can be applied to determine the expected execution time of individual activities based on the deadline for the entire workflow.

2.1.9 Comparison of Existing SWFS Approaches

Table 2.1 provides a comparison of the SWFS approaches based on the defined aspects: computing environment, optimisation method, structural representation, profitability, scheduling technique, workload type, optimisation criteria and QoS constraints.

The cost optimisation aspects of SWFS (Table 2.1) can be classified based on the computing environment aspects: (i) private cloud, (ii) public cloud, and (iii) hybrid cloud. The cost optimisation of SWFS approaches, proposed in (Pandey et al., 2010; Y. Yang et al., 2008; C. Lin & Lu, 2011), employed a private cloud environment to execute the given workflow tasks. Similarly, public cloud and hybrid cloud models have been considered by (Z. Wu et al., 2013; Bittencourt & Madeira, 2011; Abrishami & Naghibzadeh, 2012; H. Liu et al., 2011; Genez et al., 2012) in their approaches to optimally solve the SWFS problem respectively. In contrast, very few approaches proposed in (Ramakrishnan et al., 2011; Q. Wu et al., 2013; Ostermann et al., 2010) focused on utilising the strengths of both cloud computing environments to provide a more cost-effective solution for the SWFS problem. As compared to private clouds, public clouds are more expensive in terms of communication cost and execution time mainly due to the far-proximity of the resources. On the other hand, considering a hybrid model offers a highly flexible scalability feature due to the efficient utilisation of resources compared to standalone, public, and private cloud models.

Table 2.1: A comparison on cost optimisation aspects of SWFS

Reference	Computing Environment	Optimisation Method	Structural Representation	Profitability	Scheduling Technique	Workload Type	Optimisation criteria	QoS constraints
(Z. Wu et al., 2013)	Hybrid cloud	Meta-heuristic	DAG	Service consumer	Static-Dynamic	Unpredicted	Multi-objective (M/S)	Activity-level
(Abrishami & Naghibzadeh, 2012)	Public cloud	Heuristic	DAG	Service consumer	Dynamic	Predicted	Multi-objective (D/M)	Activity-level
(Ramakrishnan et al., 2011)	Cloud	Heuristic	DAG	Service provider	Dynamic	Predicted	Multi-objective (D/R/A/M)	Workflow-level
(Nargunam & Shajin, 2012)	Private & Public cloud	Greedy	N/A	Service provider	Dynamic	Predicted	Single-objective	N/A
(Q. Wu et al., 2013)	Cloud	Critical-path method	DAG	Service provider	Dynamic	Predicted	Single-objective (M)	N/A
(Ostermann et al., 2010)	Cloud	Fuzzy	DAG	Service provider	Dynamic	Predicted	Multi-objective (D/M)	N/A
(H. Liu et al., 2011)	Hybrid cloud	Heuristic	DAG	Service provider	Static	Unpredicted	Multi-objective (B/R/S)	Workflow-level
(Y. Yang et al., 2008)	Private cloud	Market-Driven	Just-in-time graph	Service consumer	Dynamic	Unpredicted	Multi-objective (B/D)	Activity-level
(Genez et al., 2012)	Public cloud	Heuristic	DAG	Service consumer	Static	Predicted	Multi-objective (D/S)	Workflow-level
(Xu et al., 2009)	Private cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (B/M)	Activity-level

Continued on next page

Table 2.1 – continued from previous page

Reference	Computing Environment	Optimisation Method	Structural Representation	Profitability	Scheduling Technique	Workload Type	Optimisation criteria	QoS constraints
(Saeid Abrishami, 2013)	Private cloud	Meta-heuristic	DAG	Service provider	Dynamic	predicted	Multi-objective (D/M)	Activity-level
(C. Lin & Lu, 2011)	Private cloud	Clustering	DAG	Service provider	Dynamic	Unpredicted	Single-objective (M)	Workflow-level
(Pandey et al., 2010)	Private cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Single6-objective (M)	Activity-level
(Tanaka & Tatebe, 2012)	Private cloud	Partitioning	DAG	Service provider	Dynamic	Unpredicted	Single-objective	Activity-level
(Bittencourt & Madeira, 2011)	Hybrid cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (D/M)	Workflow-level
(Szabo et al., 2014)	Public cloud	Meta-euristic	DAG	Service consumer and provider	Dynamic	Predicted	Multi-objective (D/R/A)	Activity-level
(Shi, 2014)	Private cloud	Dynamic programming	DAG	Service provider	Dynamic	Predicted	Multi-objective (B/D)	Workflow-level
(Viana, de Oliveira, & Mattoso, 2011)	Public cloud	Heuristic	VisTrails	Service provider	Dynamic	Unpredicted	Multi-objective (R/A/S)	Activity-level
(Malawski et al., 2012)	Public cloud	Heuristic	DAG	Service consumer	Static and Dynamic	Unpredicted	Multi-objective (B/D/R/M/S)	Activity-level
(Zhu, Wu, & Zhao, 2012)	Private cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (D/R/A/M)	Workflow-level
(X. Lin & Wu, 2013)	Public cloud	Heuristic	DAG	Service consumer and provider	Dynamic	Predicted	Multi-objective (B/D/A/M)	N/A

Legends: B = Budget; D = Deadline; R = Reliability; A = Availability; M = Makespan; S = Service level agreement

The SWFS approaches can be categorised based on different aspects of optimisation method such as heuristic (Ramakrishnan et al., 2011; Saeid Abrishami, 2012), clustering (C. Lin & Lu, 2011), critical path (Q. Wu et al., 2013), fuzzy (Ostermann et al., 2010), extended critical activity (Y. Yuan et al., 2006), holt-winter's method (Afzal, Darlington, & McGough, 2006), greedy (Nargunam & Shajin, 2012), market-driven (Y. Yang et al., 2008), meta-heuristic (Z. Wu et al., 2013; Saeid Abrishami, 2013), mathematical modeling (J. Yu, Buyya, & Tham, 2005)) and partitioning (Tanaka & Tatebe, 2012). Majority of the SWFS approaches focused on employing heuristic and meta-heuristic as an optimisation method. Meta-heuristic approaches achieved a better performance in terms of effectiveness and accuracy at the cost of extended execution time compared to heuristic methods. The structural representation parameter categorises SWFS approaches into network routing (Stevens et al., 2009), DAG (Genez et al., 2012; Prodan & Wieczorek, 2010; Q. Wu et al., 2013; Afzal et al., 2006), atomic task partitioner (Ranaldo & Zimeo, 2009), and just-in time graph (Y. Yang et al., 2008). Majority of the state-of-the-art SWFS approaches used DAG structural model to graphically visualise the dependency among the SWFS tasks, especially focusing on the cost parameters (execution time, and communication time).

The existing scheduling techniques have been classified into three parameters including static (Talukder, Kirley, & Buyya, 2009; H. Liu et al., 2011; Sakellariou et al., 2007), dynamic (Q. Wu et al., 2013; W.-N. Chen & Zhang, 2009; X. Lin & Wu, 2013), and static-dynamic (Z. Wu et al., 2013; Ranaldo & Zimeo, 2009; Afzal et al., 2006). Dynamic scheduling is efficient for a cloud computing environment due to its ability to handle the arriving tasks. The selection of workload type mainly depends on the tasks arrival rate based on the defined suitability criterion between using predicted or unpredicted mode. The workload types of SWFS approaches have been categorised into two modes, predicted mode (J. Yu & Buyya, 2006b; Q. Wu et al., 2013; Ostermann et al., 2010; W.-N. Chen &

Zhang, 2009; Y. Yuan et al., 2006; Sakellariou & Zhao, 2004b), and unpredicted mode (Z. Wu et al., 2013; Pandey et al., 2010; Xu et al., 2009; H. Liu et al., 2011; Sakellariou et al., 2007; C. Lin & Lu, 2011).

2.1.10 Discussion on Cost Optimisation Aspects

Computing environment: A large number of cost optimisation of SWFS systems (35%) implemented or developed in the private cloud are used to present Software as a Service (SaaS) issues. The advantage of utilising the SaaS cloud model for experimentation purpose is that, SaaS does not require any details about the computational infrastructure (Infrastructure as a Service) where the requests are being processed. Surprisingly, public cloud has achieved less attention (22%) from researchers compared to other environments. Moreover, it has been found that small number of models (9%) focused on minimising execution cost in the hybrid cloud. This could be due to the difficulty of calculating the total computional cost for these models owing to the heterogeneity of communication aspects and load balance challenges among resources (i.e. resource allocation, resource utilisation and resource migration). Therefore, it is of paramount importance to focus on hybrid cloud for sharing the workflow management system's workload. In contrast, it might be possible in future to propose approaches that combine the private and public (hybrid) cloud models to offer sufficient power for processing to accomplish workflow in a specified execution timeframe.

Optimisation method: it has been found that heuristic (57%), market-oriented (10%), and meta-heuristic (11%) approaches have attained major attention of researchers compared to other optimisation methods. Heuristic approaches have the highest potential to compute more accurate results. In contrast, market-oriented and meta-heuristic approaches are mainly used to achieve better performance compared to the fast heuristic methods, but with little compromise on execution time. Therefore, designing a hybrid approach (by

integrating the features of existing heuristic and meta-heuristic search algorithms) can help to improve the scalability challenge due to concurrent processing.

Structural representation: From studying the frequency of structural representations, it can be clearly found that majority of the work (85%) have considered DAG or the modified DAG model as the structural representation. The DAG is able to handle very complex cost optimisation workflow applications in cloud computing systems. It shows the precedence constraints relationship between the workflow tasks. However, only few approaches (15%) that used different alternative structural representation methods have been adopted (i.e. grid broker (6%), just in time (3%), atomic task partitioning (3%), and multiplier level architecture (3%)). For future studies, there is a need to introduce a different kind of method that is applicable to large-scale data (data-intensive) for SWFS.

Scheduling technique: From analysing the frequency of scalability aspects reported in the literature, it can be clearly found that majority of the work targeted dynamic approaches (77%). This is due to the fact that the dynamic method requires prior knowledge about the parameters. In contrast, some work (23%) has focused on the static methods for cost optimisation of SWFS approaches.

Workload type: The predicted (batch mode) type of workload remains a key focus (62%) of researchers in the cost aspect of SWFS. In contrast, some work (38%) has focused on considering the unpredicted (on-line mode) type of workload. This is due to the ability that the batch mode offers to SWFS models by maximising the throughput of the workload while minimising the turnaround time (the time between task submission and task completion).

Optimisation criteria: Most of the reviewed approaches (91%) have focused on multi-objective optimisation, while only (9%) approaches targeted single-objective based optimisation. This is mainly due to the fact that SWFS contains multiple objectives and

constraints in cloud computing environments.

2.2 Cost Optimisation Parameters of SWFS

This section critically analyses the devised classifications for cost optimisation parameters of SWFS in cloud computing. A complete discussion on the sub-classification of cost parameters including the monetary cost and temporal cost is presented in the following sub-sections. Finally, the section provides the correlations between the surveyed cost optimisation of SWFS approaches and the profitability by extracting their association with cost optimisation parameters.

After analysing the cost optimisation parameters considered by researchers in the area of SWFS in cloud computing, it is found that the classification of the cost optimisation parameters is dependent on two types: (i) monetary cost parameters, and (ii) temporal cost parameters, as shown in Figure 2.3.

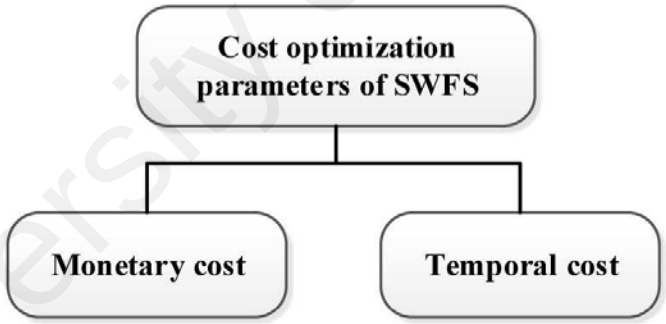

Figure 2.3: A classification of cost optimisation parameters of SWFS

The scheduling approaches are supposed to estimate in advance whether a workflow will be able to meet the requested constraints (e.g., deadline) or not (Ramakrishnan et al., 2011; Hirales-Carbajal et al., 2012). However, the estimation process may be compromised due to uncertainty in task estimations especially in the case of deadline-sensitive applications (e.g., weather forecasting). Also, the resource providers find it hard to ensure the resource availability due to the variability and complexity of the underlying resource characteristics

and access policies. Researchers have considered three parameters to overcome the aforementioned challenges: (i) Estimated Execution Time (EET); (ii) Estimated data Transfer Time (ETT); and (iii) Estimated Finish Time (EFT) (Ramakrishnan et al., 2011; Bittencourt et al., 2012; Choudhary, Kacker, Choudhury, & Vashisht, 2012; Tanaka & Tatebe, 2012; K. Liu et al., 2010; Ostermann et al., 2010; Saeid Abrishami, 2012; Rezaei, Chiew, Lee, & Shams Aliee, 2014). The scheduler needs to include these parameters in the workflow definition to enhance the estimation process by considering the historical results.

Hence, it is crucial to recognise the parameters of the monetary cost and temporal cost, and the inter-dependent parameters. The specific breakdown (sub-classifications) and details on each of the monetary cost parameters are given in Section 2.2.1 and those of the temporal cost parameters are presented in Section 2.2.2.

2.2.1 Monetary Cost Parameters

This section presents two main sub-sections including classification of monetary cost parameters and monetary cost parameters from the profitability aspect.

2.2.1.1 Classification of Monetary Cost Parameters

This section provides details on the sub-classification of monetary cost optimisation parameters as shown in Figure 2.4: (i) the estimated execution cost, (ii) the cost of service offered by the service provider, (iii) computation cost, (iii) communication cost, (iv) elasticity cost, and (v) cost of data storage.

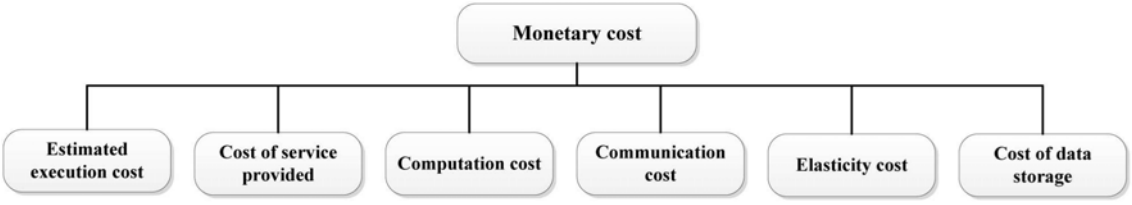

Figure 2.4: Sub-classification of monetary cost optimisation parameters of SWFS

Estimated execution cost: Estimated execution cost can be measured before the process of workflow scheduling, and it assists the algorithm with decision making for scheduling tasks and data with the help of the DAG graph partitioning of the workflow. The estimated execution cost parameter covers two main elements: (i) the estimated computation cost, and (ii) the estimated communication cost (Tanaka & Tatebe, 2012; de Oliveira, Viana, Ogasawara, Ocaña, & Mattoso, 2013; Hirales-Carbajal et al., 2012; Verma & Kaushal, 2017). The estimated execution cost represents the cost of processing a task at the resource. In contrast, the estimated communication cost refers to the cost of sending the required data along the edges of DAG from one resource to another based on the tasks' dependencies.

Cost of service provided: Cost of service provided represents the cost of the service offered by the service provider as an external cost to fulfill a service request to the service consumer, usually measured in dollars. Every service provider may have particular strategies for task-level scheduling to optimally use the system's running cost at its own data center (Z. Wu et al., 2013; Deelman, Juve, Rynge, Voeckler, & Berriman, 2013; Bittencourt & Madeira, 2013; Byun, Kee, Kim, & Maeng, 2011; Kousalya et al., 2017).

It is essential to reduce the total computional cost of application execution on the resources offered by the cloud service providers such as GoGrid and Amazon (Pandey et al., 2010). The total computional cost of service provided by the service provider can be calculated based on the cost of the services used by the workflow scheduler (Prodan & Wieczorek, 2010). Therefore, workflow execution cost is the sum of the cost of all activities (Y. Yuan et al., 2006). Additionally, the cost of an application is defined by the summation of the costs of all selected service instances (W.-N. Chen & Zhang, 2009).

Computation cost: Computation cost is defined as the cost of using computing resources and is usually measured in dollars per hour. The cost of computation is generally the user's main concern (Sharif, Taheri, Zomaya, & Nepal, 2013; Lingfang et al., 2012; Yan,

Luo, Hu, Li, & Zhang, 2013; Z. Wu, Ni, Gu, & Liu, 2010). The computation cost of tasks on the host computer is inversely proportional to the time spent on computing these tasks using the resources (Z. Wu et al., 2013; Pandey et al., 2010). For instance, the computation cost of the public cloud is represented by the amount of money to be paid for using the enterprise's computation resources, which can be categorised based on different computational specifications (Deelman et al., 2013; Pandey et al., 2010; Bhise & Mali, 2013; Byun et al., 2011).

The total computional cost of computing the workflow is also affected by the data size (Pandey et al., 2010). Thus, decreasing the execution cost of running a workflow application on the cloud system is one of the main reasons for lowering the total computional cost (Genez et al., 2012; Z. Wu et al., 2013; Saeid Abrishami, 2013; Amandeep Verma, 2012).

Communication cost: Communication cost is defined as the cost of data transferred to/from a data-storage resource, and is usually measured in dollars per megabyte of data. Communication cost between resources as well as the dependency between tasks introduces high communication cost, as data needs to be transferred from one resource to another (Varalakshmi, Ramaswamy, Balasubramanian, & Vijaykumar, 2011; Y. Wang & Lu, 2013). The communication cost is only applicable when two tasks have data dependency (Z. Wu et al., 2010). This produces higher storage and transmission cost compared to the cost of running the data (Pandey et al., 2010; J. Lin et al., 2017).

Nevertheless, the internal transfer of the data is free in many real clouds such as Amazon, so the cost of data transfer is said to be zero in this model (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013). Hence, there is no charge for data transfers within the same service provider's region (Z. Wu et al., 2010). As such, there is a significant link between the cost of data communication and data allocation. It is essential to schedule the computational tasks near the data and comprehend the moving cost of the work in

comparison to data movement (minimising the cost of communication). Also, data must be distributed over many computers, and computations must be steered towards the best place for execution in order to minimise communication cost (Genez et al., 2012; Foster et al., 2008).

Elasticity cost: Elasticity in resources provisioning to the service consumer's computing environment is one of the most important features of a cloud system. The cloud system is able to handle the execution of complex computational tasks, which require powerful resources (e.g., machines and storages). Thus, the cloud computing system is suitable to address the problems of large-scale SWFS applications (L. Mao, Yang, & Xu, 2013; Bittencourt, Senna, & Madeira, 2010; Deldari et al., 2017).

In contrast, the elasticity of SWFS is bounded by the number of resources requested by the scheduling algorithm (Bittencourt & Madeira, 2011). Therefore, the SWFS approaches need to make full use of the resources' elasticity by providing an efficient resource allocation within the lowest cost when tasks are completed earlier than the predicted time. For instance, in a hybrid cloud system, the service consumer is required to efficiently utilise the usage of public cloud resources that can be aggregated to the private resources' pool as necessary (Genez et al., 2012; Byun et al., 2011).

Cost of data storage: Cloud Workflow Management System (WfMS) is required to deliver on-demand storage services and processing power, due to the fact that the cloud WfMS has to deal with data centers which can be clusters of commodity hardware (Genez et al., 2012; Grossman, Gu, Sabala, & Zhang, 2009; D. Yuan et al., 2010; H. Yu, Bai, & Marinescu, 2005; D. Yuan, Yang, Liu, Zhang, & Chen, 2012). Executing a large size of SWFA application usually needs high performance computing resources as well as massive storage (D. Yuan et al., 2012). The execution of a workflow task consists of three phases, downloading of input data from the storage system, running the task, and

transferring output data to the storage system (Byun et al., 2011). Therefore, for a cloud WfMS storing all the data generated during workflow executions may cause a high storage cost (D. Yuan et al., 2010; D. Yuan, Yang, Liu, & Chen, 2011). In order to reduce the total computional cost of SWFS, the WfMS requires a strategy that can reduce the cost of the cloud WfMS by automatically storing only appropriate data in the cloud storage (Genez et al., 2012; D. Yuan et al., 2010, 2012; Broberg, Buyya, & Tari, 2009; Rodriguez & Buyya, 2017). Several strategies have been reported for cloud storage. For example, BigTable includes Google File System (GFS), SimpleDB data cloud, and MapReduce infrastructure, Amazon's S3 storage cloud, EC2 compute cloud and Hadoop system.

2.2.1.2 Monetary Cost Parameters from Profitability Aspect

This section highlights the results related to the approaches that represent the relationships between profitability for the service consumers and the monetary models' cost parameters of cost optimisation scheduling as shown in Table 2.2.

From the service consumers' profitability perspective (Table 2.2), some approaches have considered the estimated execution cost as a strategy for planning the scheduling before the scheduler allocates suitable resources based on their availability. Computation cost and communication cost are frequently used during the scheduling process stage. This shows that generally, the users of workflow applications are more concerned with the amount of money they need to pay for the service. However, only few models considered the cost of data storage in their approaches that is potentially due to the need for using a private (locally) storage instead of using a storage that is provided by the service provider (remotely). Therefore, the service provider needs to consider providing more flexible storage services by keeping only appropriate data in the cloud. For elasticity cost, there is a strong need for full use of the resource elasticity by providing an efficient resource allocation within the lowest cost.

Table 2.2: Monetary cost parameters from the service consumers' point of view

Approach	Estimated execution cost	Cost of service provided	Computation cost	Communication cost	Elasticity cost	Cost of data storage
(Y. Yang et al., 2008)	x					
(J. Yu et al., 2005)	x		x	x	x	x
(Sakellariou et al., 2007)		x	x	x		
(Genez et al., 2012)		x	x	x		
(Z. Wu et al., 2013)			x			
(Abrishami & Naghibzadeh, 2012)			x	x		
(Talukder et al., 2009)			x	x		
(J. Yu & Buyya, 2006b)	x		x	x		
(W.-N. Chen & Zhang, 2009)			x	x		
(Saeid Abrishami, 2012)	x		x	x		
(Genez et al., 2012)					x	
(J. Yu & Buyya, 2006a)	x		x	x		
(Shi, 2014)	x	x	x		x	x
(Viana et al., 2011)	x					x
(Malawski et al., 2012)	x			x		
(Zhu et al., 2012)		x	x	x		
(Szabo et al., 2014)	x	x	x		x	

Table 2.3 indicates the relationship between the profitability for service providers and the monetary cost parameters of cost optimisation scheduling. From the point of view of service providers' profitability (Table 2.3), it is evident that many approaches take the estimated execution cost into consideration (Tanaka & Tatebe, 2012; Saeid Abrishami, 2012; J. Yu & Buyya, 2006a; Verma & Kaushal, 2017).

The purpose of this parameter is to measure the cost of the submitted workflow tasks that is estimated by the scheduler before the process of SWFS execution starts. There is a small number of scheduling approaches dedicated to the cost of service provided as service profit. This signifies that customers find the cost of the service provided very important. In addition, the computation and communication cost parameters are widely applied to the service category as the principal parameters representative of monetary cost.

Table 2.3: Monetary cost parameters from the service providers' point of view

Approach	Estimated execution cost	Cost of service provided	Computation cost	Communication cost	Elasticity cost	Cost of data storage
(Pandey et al., 2010)			x	x	x	
(Q. Wu et al., 2013)			x			
(Ostermann et al., 2010)		x				
(C. Lin & Lu, 2011)			x			
(Sakellariou & Zhao, 2004b)			x	x		
(Bittencourt & Madeira, 2011)		x	x	x	x	
(Ramakrishnan et al., 2011)			x	x		
(Stevens et al., 2009)		x	x	x		
(Saeid Abrishami, 2013)			x			x
(Tanaka & Tatebe, 2012)	x		x	x		
(Prodan & Wieczorek, 2010)			x	x		
(Saeid Abrishami, 2012)	x		x	x		x
(J. Yu & Buyya, 2006a)	x		x	x		
(Shi, 2014)	x	x	x		x	x
(Viana et al., 2011)	x		x	x		x
(Malawski et al., 2012)	x	x		x		
(X. Lin & Wu, 2013)	x		x	x		x
(Szabo et al., 2014)	x	x	x		x	

2.2.2 Temporal Cost Parameters

This section discusses temporal cost parameters in terms of the profitability for the service consumers and service providers. The literature review points out that in order to achieve effective and efficient cost of processing the SWFA, the schedulers should minimise the total time taken for execution (makespan) to reduce the total execution cost (Pandey et al., 2010; J. Yu & Buyya, 2006b). However, the two aims (the cost of running a process on a resource and the time expected for execution) are contradictorily related (Talukder et al., 2009; J. Yu et al., 2005; Prodan & Wieczorek, 2010; J. Yu & Buyya, 2006a). In addition, the time taken for execution and cost of execution are the two normal restrictions in the pay-per-use model of cloud computing (Xue & Wu, 2012). Thus, faster resources are more costly and vice versa with slower resources. As a result, the scheduler has to face a time-cost tradeoff in choosing suitable resources. Besides, the cost of execution rises

during longer delays, as the scheduler switches the balanced tasks to more costly services to finish off the balance of execution within the subscribed deadline.

Besides, to calculate the cost of SWFS, it is significant to take into consideration all the time intervals that each resource is utilising for task processing and data transmission (Bittencourt & Madeira, 2011; J. Yu & Buyya, 2006b; Prodan & Wieczorek, 2010; Rodriguez & Buyya, 2017).

2.2.2.1 Classification of Temporal Cost Parameters

This section categorises the temporal cost according to the scheduling stages (i.e. pre-scheduling, during scheduling, and post-scheduling) as per needs of the scheduler.

From the sub-classification of cost optimisation parameters of temporal cost as depicted in Figure 2.5, three scheduling stages are incorporated in the temporal cost: (i) pre-scheduling, (ii) during scheduling, and (iii) post-scheduling (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013). The pre-scheduling stage covers several parameters including the earliest start time (Bittencourt & Madeira, 2011; Ramakrishnan et al., 2011; Bittencourt et al., 2012; Xue & Wu, 2012; Stevens et al., 2009; Saeid Abrishami, 2013), earliest finish time (Stevens et al., 2009; Saeid Abrishami, 2013), estimated data transfer time (Saeid Abrishami, 2012), estimated execution time (Ranaldo & Zimeo, 2009; Saeid Abrishami, 2013, 2012), estimated finish time (Bittencourt & Madeira, 2011; Bittencourt et al., 2012; Ranaldo & Zimeo, 2009), latest start time, and latest finish time (Abrishami & Naghibzadeh, 2012; K. Liu et al., 2010; Saeid Abrishami, 2012; Tan, Sun, Li, Lu, & Wang, 2013). However, the during-scheduling stage includes the computation time, communication time, spare time, and ready time parameters. On the other hand, the post-scheduling stage consists of the actual start time as well as actual finish time parameters.

A- Pre-scheduling stage

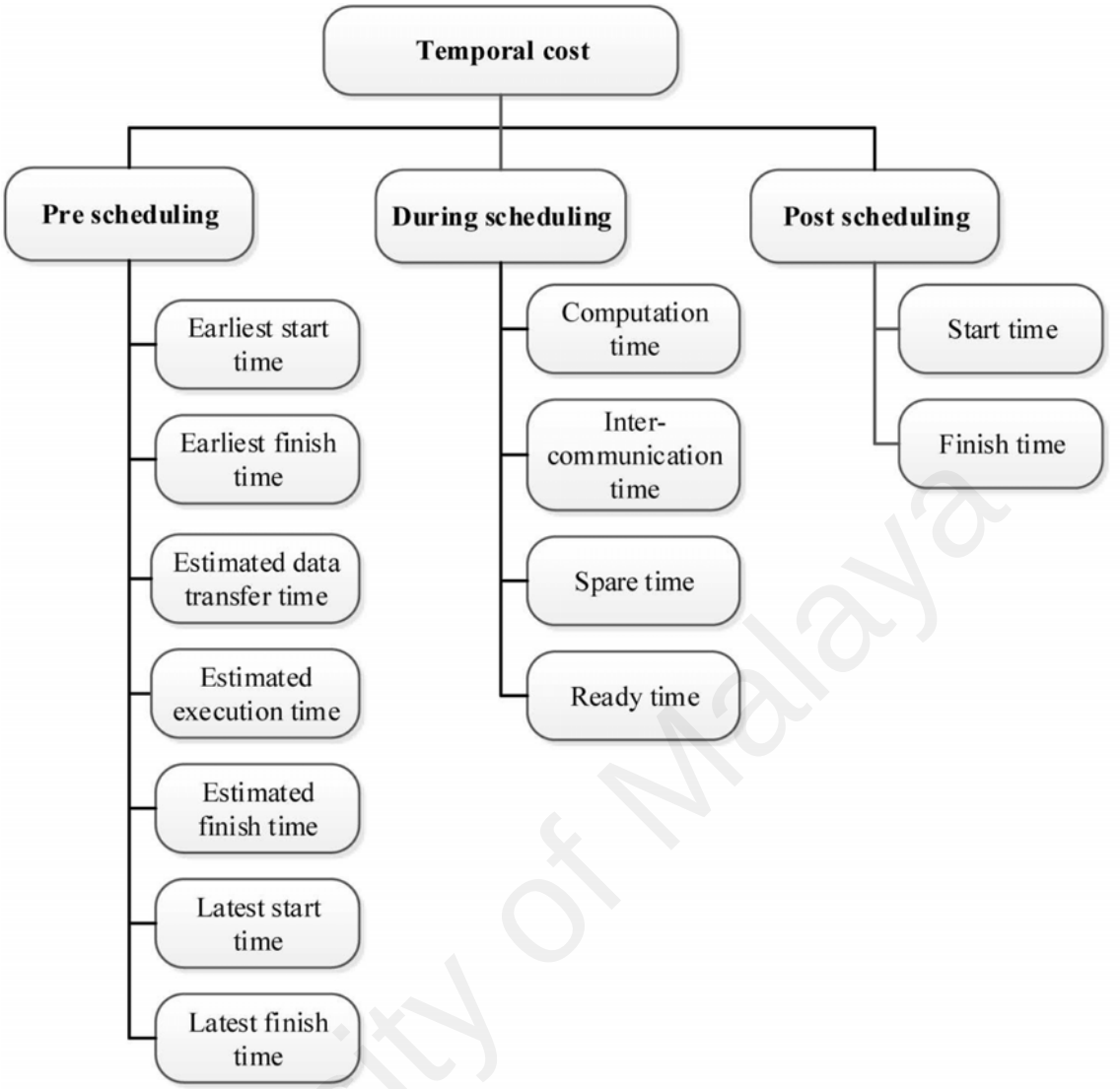

Figure 2.5: Sub-classification of temporal cost optimisation parameters of SWFS

The following are the main parameters that need to be calculated before the scheduling process to ultimately help the scheduler with scheduling decisions by estimating the temporal cost.

Earliest start time (EST): EST is defined as the earliest time to begin task computation, regardless of the actual resource to process the task that can be decided on while scheduling (Stevens et al., 2009; Saeid Abrishami, 2012). Nevertheless, it is impossible to exactly measure EST in a heterogeneous environment, as a specific cloud's computation time of tasks differs within each resource (Saeid Abrishami, 2012; Verma & Kaushal, 2017).

Every task has a period and should not be scheduled earlier than EST, and must end latest by the Finish Time (Ramakrishnan et al., 2011). The EST of each unscheduled task is described in the following equation (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013, 2012)):

$$EST(t_{entry}) = 0 \tag{2.1}$$

The task t_{entry} refers to the beginning of the workflow.

$$EST(t_i) = max_{t_p \in t_i Parents} \left\{ EST(t_p) + MET(t_p) + TT(e_{p,i}) \right\} \tag{2.2}$$

where the minimum execution time of a task t_i, $MET(t_i)$, refers to the task's execution time on a resource $r_j \epsilon R$ which has the minimum $ET(t_i, r_j)$ among all the available resources. ET denotes the estimated execution time of t_i, and the t_p is the parent task of t_i, and $e_{p,i}$ refers to the edge between the parent task node to the t_i task node in DAG. TT denotes the estimated data transfer time.

Earliest finish time (EFT): EFT for each unscheduled task is the earliest time the task's computation can finish (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013). Thus, it is essential to first calculate the EST, and then calculate the EFT for each task in the workflow prior to assigning it to the fastest resource (Saeid Abrishami, 2013). EFT can be calculated with the following equation :

$$EFT(t_i) = EST(t_i) + MET(t_i) \tag{2.3}$$

where the $MET(t_i)$ denotes the minimum execution time of a task t_i.

Estimated data transfer time (TT): TT can be defined as the amount of data that needs

to be transmitted along with the data latency and bandwidth between services, which can be used to estimate the time it takes to transfer the required data. For instance, $TT(e_{i,j})$ is defined as the data transfer time of the selected resource for t_i and t_j. Thus, TT represents the transfer cost of sending the required data along $e_{i,j}$ from resource r (processing task t_i) to resource n (processing task t_j) (Saeid Abrishami, 2012; Park et al., 2017).

Estimated execution time (ET): ET refers to the computation time for every task in each resource according to the scheduler's estimation after initiating a job request execution (K. Liu et al., 2010; Saeid Abrishami, 2012). ET is usually influenced by a few parameters (budget, total number of atomic tasks, tested configuration, and deadlines) (Ranaldo & Zimeo, 2009). Additionally, the ET for every resource differs based on task size (Ranaldo & Zimeo, 2009). An application is able to provide an estimated execution time according to the available metadata of user requests, unlike resource services (J. Yu & Buyya, 2006b).

Estimated finish time (ESHT): ESHT refers to the estimated completion time of task computation by a particular resource. Every task is scheduled to the resource with the lowest cost and earliest ESHT (Nargunam & Shajin, 2012; Bittencourt et al., 2012). ESHT can be calculated as follows (Bittencourt & Madeira, 2011)):

$$ESHT(t_i, r_k) = EST(t_i, r_k) + w(t_i, r_k) \tag{2.4}$$

ESHT (t_i, r_k) represents the estimated finish time of task i in resource k, and, $w(t_i, r_k)$ represents the execution time of task i in resource k.

Latest start time (LST): LST represents the difference between the latest finish time and estimated computation time of the task (Tan et al., 2013).

$$LST(t_i) = LFT(t_i) - D_{t_i} \tag{2.5}$$

where $LST(t_i)$ denotes the latest start time, $LFT(t_i)$ denotes the latest finish time, and D_{t_i} is the estimated task duration.

Latest finish time (LFT): LFT represents the latest time for finishing a computation task. LFT is beneficial in calculating the required time for completing a workflow with respect to the user's determined deadline of a specific set of tasks (Abrishami & Naghibzadeh, 2012; Xue & Wu, 2012; Saeid Abrishami, 2013; K. Liu et al., 2010). Thus, the LFT is an important algorithm component as it receives a workflow as the input and attempts to seek the schedule that minimises cost, reduces the total computional cost of workflow and completes the task before the LFT. LFT can be calculated using the following equation (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013)):

$$LFT(t_{exit}) = DD \tag{2.6}$$

where task t_{exit} refers to the ending of the workflow, and DD is the user-defined deadline.

$$LFT(t_i) = min_{t_c \in successors\ of\ t_i} \left\{ LFT(t_i) - ET(t_c, SS(t_i)) - TT(e_{i,j}) \right\} \tag{2.7}$$

$SS(t_i)$ is defined as the resource selected for processing t_i during scheduling. ET denotes the estimated execution time and TT denotes the estimated data transfer time.

B- During scheduling stage

The following are the main parameters to be calculated during the scheduling process.

Computation time: The computation time represents the time that is required by the computational resources to execute the workflow tasks. The main factor for every single resource is deciding the execution cost of the task's processing time (Deelman et al., 2005). Thus, it is up to the users to select the most appropriate processing budget and time (Y. Yuan et al., 2006). Workflow execution comprises the running time of the tasks and

data transfer in and out of the computation resource (Deelman et al., 2005). Note that, communication cost and computation cost are inversely proportional to communication time and execution time, respectively (J. Yu & Buyya, 2006b).

Communication time: Data transfer from one computer source to another is time consuming and the duration of time is dependent on the amount of data that needs to be transferred between the corresponding tasks. Moreover, it is not dependent on the services that execute them (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013; Menasce & Casalicchio, 2004). The time for data transmission is dependent on the selected services and the service provider's bandwidth (Saeid Abrishami, 2012). Nevertheless, the time for data transfer between two arbitrary tasks is constant and not dependent on the selected services (Abrishami & Naghibzadeh, 2012). Therefore, if all workflow tasks are scheduled at the same instance, the time for data transfer between them becomes zero, but the time for data transfer outside of the tasks should be still taken into consideration (Saeid Abrishami, 2013).

Spare time: Spare time (also referred as Application Spare Time) represents the time difference between the expected finish time (makespan) of the initial schedule and the deadline defined by the user for the whole workflow (Zheng, 2010; Byun et al., 2011). The distribution scheme of the spare time affects the overall cost (Byun et al., 2011). In order to guarantee the feasibility of the workflow execution when the actual execution time of task changes to a certain extent from the predicted time, the spare time is assigned to each workflow task based on its deadline. In the literature, two main approaches, including critical-path-based allocation and recursive allocation have been proposed for spare time allocation (Wieczorek, Hoheisel, & Prodan, 2008; H. Zhao & Sakellariou, 2007).

Ready time: Ready time is defined as the earliest time for the first task to be executed and this task is computed based on the parent tasks (J. Yu et al., 2005). The following

equation is used to calculate the ready time of task t_i (J. Yu & Buyya, 2006a)):

$$readyTime(t_i) = max_{t_j \in p_i} endTime(t_j) \tag{2.8}$$

where p_i is the set of parent tasks of t_i, and endTime (t_j) denotes the time required to end the execution of task t_j (deadline).

C- Post scheduling stage

The workflow manager examines the workflow by consulting the repository or database containing the information linked to cost and performance records. The repository may also contain historical information regarding the execution of past services requested by SaaS clients and all their resource performance (Genez et al., 2012). Thus, the user's input on cost and time will be considered for scheduling the next time around to ensure user satisfaction (Y. Yang et al., 2008). A major consideration in the execution of applications that are performance-driven is effective scheduling, such as cost-driven and dynamic workflow environments like a cloud (Amandeep Verma, 2012; J. Lin et al., 2017).

Performance estimation for resource services is derived by utilising current performance estimation techniques, for instance, historical (Ranaldo & Zimeo, 2009; Bjorkqvist, Chen, & Binder, 2012)) and empirical data (Abrishami & Naghibzadeh, 2012), and analytical modeling (Saeid Abrishami, 2013)) to predict the time taken for task execution on each discovered resource service (Bittencourt & Madeira, 2011; Abrishami & Naghibzadeh, 2012; Ranaldo & Zimeo, 2009; J. Yu et al., 2005; J. Yu & Buyya, 2006b; Ostermann et al., 2010; Afzal et al., 2006; J. Yu & Buyya, 2006a; Verma & Kaushal, 2017). In addition, the costs of communication and computation for workflows are calculated from historical data of past filtered executions (Bittencourt & Madeira, 2011).

Start time (actual): Every task has four components: (i) *serviceID*, (ii) *taskID*, (iii)

endTime, and (iv) *startTime*. *serviceID* and *taskID* identify where each task is assigned to which resources. *startTime* and *endTime* represent the allocated time frame on the resource for task execution (Stevens et al., 2009; J. Yu & Buyya, 2006b; Sakellariou & Zhao, 2004b). The entire workflow completes based on parallel and serial constraints between the start and finish times (Y. Yuan et al., 2006).

Once all tasks are scheduled, each task has a start time that is measured using the deadlines of the parent tasks in the workflow (Abrishami & Naghibzadeh, 2012). There are two concepts of tasks' start times in the scheduling algorithms. The first concept is, supposing the start time is EST, which is calculated prior to the workflow being scheduled; however, the real start time concept is calculated after scheduling the tasks (Saeid Abrishami, 2013, 2012). It is helpful to compare the start time estimated statically and the minimal spare time saved to determine rescheduling in the future (Sakellariou & Zhao, 2004b).

Finish time (actual): This is the time actually used to complete task execution (Sakellariou et al., 2007; Q. Wu et al., 2013; Sakellariou & Zhao, 2004b; Park et al., 2017).

2.2.2.2 Temporal Cost Parameters from Profitability Aspect

This section presents the results related to the relationship between the service consumers of the cost optimisation approaches and their temporal cost parameters (Table 2.4).

From the point of view of service consumers' profitability (Table 2.4), several approaches focus on measuring pre-scheduling parameters due to the significance of determining the estimated execution time, which is required to fulfil the customer's QoS attributes (i.e., deadline and makespan). Similarly, the monetary cost category, with computation time and communication time, is extensively used during the scheduling process stage. Also, several approaches measure the pre-scheduling stage. This shows that workflow application users are more concerned about the waiting duration needed for the service to be accomplished

by resources. Only few approaches have considered the ready time and spare time.

Table 2.4: Temporal cost parameters from service consumers' point of view

	Pre-Scheduling							During Scheduling					Post Scheduling	
Approach	**Latest Start Time (LST)**	**Latest Finish Time (LFT)**	**Earliest Start Time (EST)**	**Earliest Finish time (EFT)**	**Estimated Finish Time (EFHT)**	**Estimated Execution Time (ET)**	**Estimated data transfer time (TT)**	**Ready Time (RT)**	**Spare Time**	**Resource Allocation time**	**Computation Time**	**Communication Time**	**Start Time**	**Finish Time**
(Y. Yang et al., 2008)						x								
(J. Yu et al., 2005)			x			x		x			x	x	x	
(Sakellariou et al., 2007)				x									x	x
(Z. Wu et al., 2013)			x	x										
(Abrishami & Naghibzadeh, 2012)		x	x	x								x	x	
(Talukder et al., 2009)				x								x		
(J. Yu & Buyya, 2006b)						x					x	x	x	
(W.-N. Chen & Zhang, 2009)			x										x	
(Saeid Abrishami, 2012)		x	x			x	x				x	x	x	
(J. Yu & Buyya, 2006a)				x				x			x	x	x	
(Shi, 2014)	x					x	x				x	x	x	
(Zhu et al., 2012)		x	x	x			x			x	x	x	x	x
(Szabo et al., 2014)											x	x		x

Table 2.5 shows the results related to the relationship between the service providers of cost optimisation approaches and their temporal cost parameters.

Table 2.5: Temporal cost parameters from the service providers' point of view

	Pre-Scheduling							During Scheduling					Post Scheduling	
Approach	**Latest Start Time (LST)**	**Latest Finish Time (LFT)**	**Earliest Start Time (EST)**	**Earliest Finish time (EFT)**	**Estimated Finish Time (EFHT)**	**Estimated Execution Time (ET)**	**Estimated data transfer time (TT)**	**Ready Time (RT)**	**Spare Time**	**Resource Allocation time**	**Computation Time**	**Communication Time**	**Start Time**	**Finish Time**
(Nargunam & Shajin, 2012)			x		x						x			
(Ranaldo & Zimeo, 2009)						x					x	x	x	
(Q. Wu et al., 2013)		x	x										x	x
(Y. Yuan et al., 2006)											x		x	
(C. Lin & Lu, 2011)			x	x				x						
(Sakellariou & Zhao, 2004b)									x		x	x	x	x
(Bittencourt & Madeira, 2011)			x		x							x		
(Ramakrishnan et al., 2011)		x	x							x	x	x	x	
(Stevens et al., 2009)			x	x							x	x	x	

(Saeid Abrishami, 2013)		x	x	x		x						x	x	
(Tanaka & Tatebe, 2012)				x							x	x		
(Afzal et al., 2006)												x		
(Saeid Abrishami, 2012)		x	x			x	x				x	x	x	
(J. Yu & Buyya, 2006a)				x				x			x	x	x	
(Shi, 2014)	x					x	x				x	x	x	
(Viana et al., 2011)											x	x		
(Malawski et al., 2012)	x		x				x				x	x	x	x
(X. Lin & Wu, 2013)	x	x	x	x			x			x	x	x	x	x
(Szabo et al., 2014)											x	x		x

From the point of view of service providers' profitability (Table 2.5), most of the approaches focus on calculating the pre-scheduling parameters. Therefore, it is crucial for the service providers to identify the required execution time to schedule tasks to the available resources. During the scheduling process stage, researchers have given more attention to determine computation time and communication time parameters. Thus, the service consumer is concerned about the waiting period required for resources to accomplish the service. Similarly, in the service consumers' category, there are not many approaches that consider the ready time and spare time.

2.2.3 Discussion on Cost Optimisation Parameters

From analysing the monetary cost parameters from the point of view of service consumers' profitability and service providers' profitability, surprisingly, few approaches have considered the estimation of execution cost parameter. Yet, estimated execution cost is one of the important challenges that requires the scheduler in handling the uncertainties of the input of SWFS algorithms. The majority of the proposed approaches emphasise cost in their approaches. This shows that there is a strong dependency between computation cost, communication cost and the total monetary cost.

During the scheduling stage, several models consider the cost of a service offered by service providers. This signifies that customers consider the cost of the service provider highly important, which must be calculated as an external cost to fulfil a service request

to the service consumers. Only few models consider the cost of data storage in their approaches, which is potentially due to private resource usage instead of using a service offered by service providers. A large amount of work is to be expected to optimise data storage parameter due to recent focus on the big data challenge. Similarly, future researchers can utilise the resource elasticity (which plays a strong role in cloud) by providing an efficient resource allocation within the lowest cost.

From analysing the temporal cost parameters according to the point of view of service consumers' and service providers' profitability, several approaches have focused on measuring pre-scheduling parameters due to the significance of determining the estimated execution time, which is required by the scheduler for handling the uncertainties of the input challenge for SWFS algorithms. At the same time, it is very important for the service providers to calculate the required execution time to schedule tasks to the available resources.

As during the scheduling process stage, the computation time and communication time parameters are extensively used. This shows that workflow application service consumers are more concerned about the time needed for the service to be accomplished by resources. This also highlights that there is a direct relationship between computation time and communication time parameters. So, in order to examine cost performance of the cost optimisation of SWFS algorithms, researchers should consider this relationship. From the pre-scheduling stage, there are not many approaches that consider the ready time and spare time. Thus, the scheduling approaches in the future study should adjust to the aforementioned cost optimisation parameters for the execution time of workflow process model.

Regarding the response time challenge in cloud, traditionally, cloud systems aim to achieve better trade-off between performance (i.e. response time) and cost. The value of

the response time (the duration of a service between calling and return) is specified in the service level agreement. In most cases, the service consumers of workflow application want fast response times regardless of cost, application owners want fast response times without spending too much money, and service providers seek to reduce the cost of running all applications within their service agreements, regardless of ownership (Dutta & VanderMeer, 2011). Thus, the challenge faced by a service provider is how to minimise the cost while maintaining the resource utilisation and low service response time (Bjorkqvist et al., 2012; Garg, Buyya, & Siegel, 2010; Y. Yuan et al., 2007). SWFS approaches are required to consider achieving the application's submission time which is equal to the application execution start time. For the aforementioned reasons, cloud service providers usually use the auto-scaling mechanism to minimise the response time of customer requests to improve the user experience (M. Mao & Humphrey, 2011; Y. Yuan et al., 2009).

2.3 Cost Optimisation Challenges of SWFS

Through an extensive literature review, the challenges of SWFS have been classified into three main types (Figure 2.6): (i) QoS performance; (ii) system functionality; and (ii) system architecture. The classification of these challenges is based on the WfMS issues proposed by Liu (2012) (X. Liu, 2012) and the standard reference model of workflow scheduling proposed by the Workflow Management Coalition (WfMC) (N. Kaur et al., 2011; Coalition, 2005; Rodriguez & Buyya, 2017).

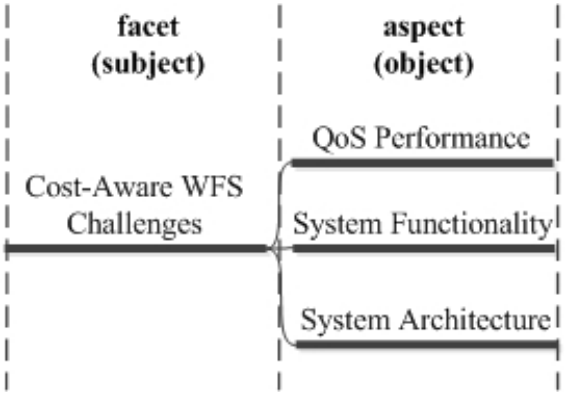

Figure 2.6: The SWFS challenges

Thus, the relevant sub-taxonomies of each of the SWFS challenges regarding QoS performance will be discussed in Section 2.3.1 based on the associations with SWFS approaches. Also, the relevant sub-taxonomies of each of the SWFS challenges regarding system functionality and system architecture will be discussed in Sections 2.3.2 and 2.3.3 respectively, based on the associated SWFS approaches.

This section critically analyses the devised taxonomies for SWFS challenges in cloud computing environment. Firstly, it presents the sub-taxonomy of challenges. Then, it depicts the correlation of these challenges with key aspects of WfMS in cloud computing. Finally, it provides a grouping of reviewed models based on the profitability by extracting its association with challenges.

2.3.1 Quality of Service (QoS) Challenges

The QoS constraints have a major effect on SWFS in cloud computing, since the success of computation tasks is highly dependent on the desired QoS levels, which are: quality of results, execution time, throughput, reliability, monetary cost, deadline, trust, and budget (Ranaldo & Zimeo, 2009; X. Liu, 2011; T. Chen, Bahsoon, & Theodoropoulos, 2013). In the literature, two major SWFS types have been devised (K. Liu et al., 2010; J. Yu & Buyya, 2006b; Lingfang et al., 2012; Tilak & Patil, 2012; W.-J. Wang, Chang, Lo, &

Lee, 2013): (i) Best effort-based scheduling – which attempts to minimise execution time without considering other factors such as the monetary cost of accessing resources and various users' QoS satisfaction levels; and (ii) QoS constraint-based scheduling – which attempts to maximise performance under QoS and other system performance constraints such as execution cost, time minimisation under budget constraints, or cost minimisation under deadline constraints. On the other hand, according to standard ISO 8402 (?, ?), ISO/IEC 13236:1998, and UTI (ITU. Recommendation, n.d.), QoS may be defined in terms of purpose of workflow service to be provided to consumers. Therefore, several QoS constraints must be taken into consideration for a given service when designing an efficient WfMS in a cloud computing environment (H. Liu et al., 2011; Varalakshmi et al., 2011; Saeid Abrishami, 2012; Anbazhagan Mani, 2002). This section depicts the major QoS performance challenges as depicted in Figure 2.7, considered by the reviewed approaches with relevance to the generic QoS framework (X. Liu, 2012).

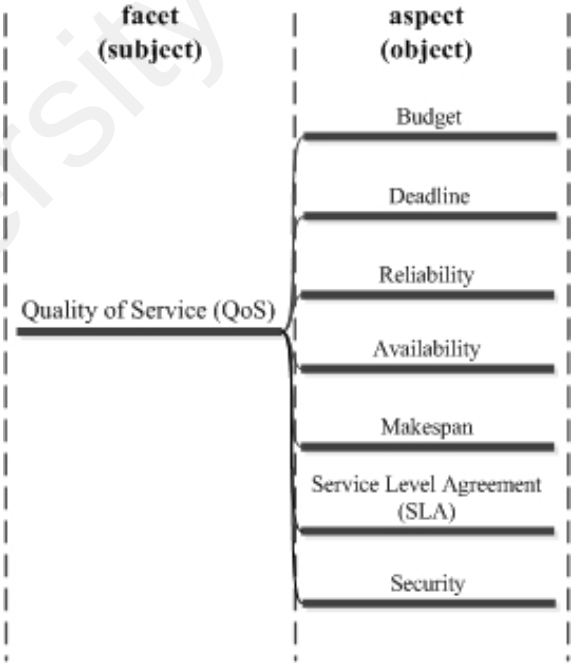

Figure 2.7: Sub-taxonomy of QoS performance challenges

2.3.1.1 Types of QoS Challenges

In the literature, a number of QoS cost-based strategies for SWFAs have been proposed, which are intended to provide maximum customer satisfaction. Figure 2.8 shows the interaction between the QoS challenges and how these challenges affect the cost. The following sub-section presents the reported QoS challenges for SWFS.

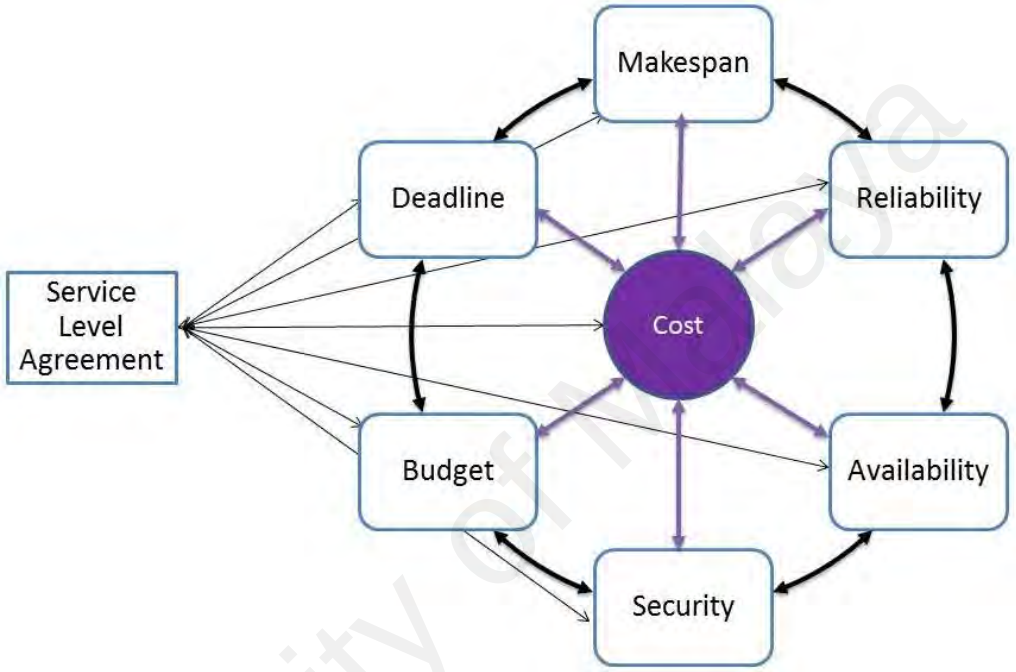

Figure 2.8: Interaction among QoS challenges

Budget: In the context of QoS, budget can be defined as the cost threshold that a user wants to pay a service provider to avail the desired services (Saeid Abrishami, 2013; H. Liu et al., 2011; Sakellariou et al., 2007). Budget is dependent on the deadline selected by the user to offer maximum QoS at minimum cost (Z. Wu et al., 2010; Choudhary et al., 2012; Ranaldo & Zimeo, 2009; Bjorkqvist et al., 2012; Tsai et al., 2014) (Figure 2.8). Furthermore, the budget constraint of SWFS from the service providers' view maps every task onto a suitable service to minimise workflow execution time and complete it within budget (J. Yu & Buyya, 2006b). Therefore, the overall goal of the scheduling process is to achieve a schedule that offers a set of resources using the specified budget (J. Yu &

Buyya, 2006b; Sakellariou et al., 2007). For instance, the task execution cost rises during longer delays, as the scheduler switches the balance tasks to more costly services to finish the execution balance within the subscribed deadline (Rahman et al., 2011; Sharif et al., 2013; Yan et al., 2013; J. Yu et al., 2005; Gao, Ma, Zhang, Kong, & Wei, 2013; M. Mao & Humphrey, 2011; Zheng & Sakellariou, 2013).

Deadline: The time required for all workflow computations is defined as the workflow deadline (Grounds, Antonio, & Muehring, 2009; Xue & Wu, 2012; Saeid Abrishami, 2012; Y. Yuan et al., 2009). In most cases, users only specify a deadline for the entire workflow execution (J. Yu et al., 2005). Since there is a strong dependency between completion time and the required resources, a trade-off is required between the estimated completion time and required resources to judiciously perform the SWFS. For example, an earlier completion time than specified by the user requires more resources (Xue & Wu, 2012; Chunlin & Layuan, 2006; Saeid Abrishami, 2012). Thus, a deadline must be assigned to each workflow, which should be equal to or greater than the scheduling makespan of the same workflow (Abrishami & Naghibzadeh, 2012) (Figure 2.8). Consequently, if the execution time of all tasks is lower than the deadline, the cost weight changes incrementally till the execution time of all tasks meets the deadline (J. Yu et al., 2005; Tan et al., 2013; Bittencourt & Madeira, 2011). Note that the execution time plays an important role in determining the deadline value. Researchers (J. Yu & Buyya, 2006b) have determined the overall deadline by dividing the tasks to their average computation and communication time.

Reliability: Reliability is one of the main features of cloud computing that can manage independent resource allocation (Bittencourt & Madeira, 2011). Reliability refers to the probability that a task can successfully complete the assigned workflow (Chunlin & Layuan, 2006). The user may require a specific level of reliability based on the assigned

workflow, which ultimately reduces the chances of failure to complete that workflow. The reliability preference shows that the tasks are scheduled to the resource instances with higher reliability and can be measured on a scale from 0 to 1 according to the resource's degree of reliability (Varalakshmi et al., 2011). Therefore, the service reliability should be considered, as highly unstable resources may take longer execution time (and cost) compared with more reliable alternatives (Prodan & Wieczorek, 2010) (Figure 2.8).

Availability: WfMS availability can be increased by focusing on improving testability and maintainability. In the SWFS process, the low cost of services may cause high computation and shortage of resources for performing workflow execution. The waiting time of resources (to become available) can create an unnecessary and critical delay (Srinivasan, 2012; J. Yu et al., 2005). Unavailability negatively impacts the overall cost of SWFS. In the literature, two main scenarios have been reported that highlight this impact: (i) cost attached to communication delay, known as additive float cost; (ii) the CPU's availability at the end node's computing element, which is known as the cost of bounded integer (Stevens et al., 2009), and (iii) reliability of the resources that needs to be provided by service providers (Manvi & Krishna Shyam, 2013; Albodour, James, & Yaacob, 2012)) (Figure 2.8).

Makespan: Makespan can be defined as the overall time to execute the whole workflow by considering the finish time of the last completed task (Saeid Abrishami, 2012; Q. Wu et al., 2013). Obviously any delay in the task's execution time will have an effect on the makespan of workflow application (Sakellariou & Zhao, 2004b). In a traditional SWFS, users generally choose to minimise the workflow makespan of the tasks (Abrishami & Naghibzadeh, 2012; Y. Yuan et al., 2006). Thus, the reported cloud-based SWFS approaches focus on how to assign tasks to resources so that the precedence constraints are retained while the makespan is minimised (Sakellariou & Zhao, 2004b). Hence, the

major aim of makespan is to reduce the execution cost within a lower completion time while meeting the users' requirements effectively (Tanaka & Tatebe, 2012; Z. Wu et al., 2010; Y. Yuan et al., 2006; J. Lin et al., 2017) (Figure 2.8).

Service Level Agreement (SLA): A legal document that considers the perspectives of both service consumers and service providers to formally describe, and deliver the cloud services is known as SLA (Genez et al., 2012; Saeid Abrishami, 2012). These perspectives include explanations of QoS delivery performance guarantees. Therefore, the SLA is related to all the QoS dimensions which must be mentioned by the service provider in order to fulfill the service consumer's requirement (X. Liu, 2012) (Figure 2.8). Similarly, for SLA cloud services, consumers can negotiate with service providers on the desired QoS and pricing by describing the required parameters for QoS on a pay-per-use basis (Abrishami & Naghibzadeh, 2012). However, SLA usually defines cost penalties when the terms for SLA are violated (e.g., a deadline is missed) (Grounds et al., 2009).

Security: Security in WFS process refers to the confidentiality of workflow tasks execution and acts as a measure to determine the degree of trustworthiness of candidate resources. By ensuring the security of potential resources certainly decreases the possibility of being attacked and damaged to a great extent. Similarly, security is considered as an important quality aspect of the cloud service that ensures their confidentiality by authenticating the parties involved, encrypting messages, and providing access control. Security plays an important role in distributed computing systems to ensure the QoS of cloud workflow systems. However, security might need extra time and cost to achieve the desired objective. Therefore, the scheduler considers QoS-cost tradeoffs between different QoS parameters (e.g. availability, reliability, makespan, etc.) to optimally satisfy the QoS requirements for both service providers and service consumers.

2.3.1.2 QoS Challenges from WfMS in Cloud Computing

This section correlates two important aspects of cloud system (i.e., cloud system model and service driven related aspect) on each of the QoS challenges of SWFS. It can be clearly seen from Table 2.6 that all QoS performance challenges have been fully considered in cloud system model and cloud service driven related aspects. Ultimately, it highlights the crucial role of QoS in improving the performance of WfMS in cloud computing.

Table 2.6: Correlation between QoS performance challenges and WfMS in cloud computing

	System structure			Service driven		
Challenges	**SaaS**	**PaaS**	**IaaS**	**Service consumer**	**Service provider**	**Utility Market-oriented**
Budget	x	x	x	x	x	x
Deadline	x	x	x	x	x	x
Reliability	x	x	x	x	x	x
Availability	x	x	x	x	x	x
Makespan	x	x	x	x	x	x
SLA	x	x	x	x	x	x
Security	x	x	x	x	x	x

2.3.1.3 QoS Challenges form Profitability Aspect

Table 2.7 depicts different service consumer-related approaches with target QoS performance challenges. From the service consumers' profitability view (Table 2.7), users normally provide the workflows along with more than one QoS constraint expectation for computation purposes (Amandeep Verma, 2012). The customer pays more attention to the workflow cost and time, which are necessary to complete the execution in a more optimal manner (Y. Yuan et al., 2006).

Table 2.7: QoS performance challenges from a service consumer profitability view

		QoS performance challenges						
Profitability	Models	Budget	Deadline	Reliability	Availability	Makespan	SLA	Security
Service customer approaches	CTC (K. Liu et al., 2010)		x			x		
	Transaction Intensive Cost-constraint Algorithm (Y. Yang et al., 2008)	x	x					
	MDP (J. Yu et al., 2005)		x					
	DAG-LOSS & DAG-GAIN (Sakellariou et al., 2007)	x				x		
	ILP (Genez et al., 2012)		x					
	Cost Optimisation and Time Optimisation Scheduling Policies (Salehi & Buyya, 2010)	x	x			x		
	Market-oriented hierarchical scheduling strategy (Z. Wu et al., 2013)					x		
	SC-PCP (Abrishami & Naghibzadeh, 2012)		x			x		
	MODE (Talukder et al., 2009)	x	x			x		
	Pegasus framework (Deelman et al., 2005)				x	x		
	Genetic Algorithm (J. Yu & Buyya, 2006b)	x	x		x	x		x
	ACS (W.-N. Chen & Zhang, 2009)	x	x	x		x		x
	PCP (Saeid Abrishami, 2012)		x			x		
	Budget Constraint Based workflow scheduling (J. Yu & Buyya, 2006a)	x	x					x
	Time and Cost-constrained scheduling strategy (Ranaldo & Zimeo, 2009)							x
	BGQoS (Albodour et al., 2012)			x	x			
	TSWFS (Tan et al., 2013)	x	x	x				x

Table 2.7 indicates that a significant number of approaches focus on budget and deadline mainly due to the adaptability of these challenges in SLA and market-oriented aspects of cloud computing (Amandeep Verma, 2012). Although makespan generally depends on the computation cost of SWFS and this fact makes makespan widely considered by service consumers to determine the required speed of a cloud service. However, only two approaches (J. Yu & Buyya, 2006b) specifically focus on resource availability challenges. Similarly, three approaches (W.-N. Chen & Zhang, 2009; Tan et al., 2013; Albodour et al., 2012; Deldari et al., 2017) determine reliability.

It is clear that the majority of reviewed approaches lack in considering reliability, availability, and SLA for service consumer profitability. Consequently, future work needs to consider these for providing enhanced quality solutions. To conclude, service consumers require different cloud service types. For instance, some customers may need storage, computational, high reliability, or low-cost cloud types of services.

Table 2.8 shows the correlation results between various service providers and approaches with targeted QoS performance challenges.

Table 2.8: QoS performance challenges from the service provider's profitability perspective

		QoS performance challenges						
Profitability	Model name	Budget	Deadline	Reliability	Availability	Makespan	SLA	Security
Service provider approaches	CMSA (Grounds et al., 2009)		x				x	
	HCOC & MDP (Bittencourt et al., 2012)		x			x		
	Time and Cost-constrained scheduling strategy (Ranaldo & Zimeo, 2009)	x	x					x
	PSO (Pandey et al., 2010)					x		
	RDPSO (Z. Wu et al., 2010)	x				x		
	CPPS (Q. Wu et al., 2013)					x		
	Dynamic Resource Provisioning Techniques (Ostermann et al., 2010)					x		
	Time-Cost trade-off workflow scheduling algorithm (Y. Yuan et al., 2006)		x			x		
	SHEFT (C. Lin & Lu, 2011)					x		
	Heuristic designing Scheduling framework (Menasce & Casalicchio, 2004)					x	x	
	BMCT (Sakellariou & Zhao, 2004a)					x		
	Low-Cost Rescheduling Policy (Sakellariou & Zhao, 2004b)					x		
	HCOC (Bittencourt & Madeira, 2011)		x			x		
	WORDS (Ramakrishnan et al., 2011)		x	x	x	x		x
	GHPSO (Xue & Wu, 2012)		x					
	IC-PCPD2 & IC-PCP (Saeid Abrishami, 2013)		x			x		
	MCGP (Tanaka & Tatebe, 2012)					x		
	MQMW (Xu et al., 2009)					x		
	ACO (H. Liu et al., 2011)	x		x			x	x
	DCA (Prodan & Wieczorek, 2010)			x	x			
	MQRS (Chunlin & Layuan, 2006)	x	x	x		x		
	MINLP (Afzal et al., 2006)		x					
	PCP (Saeid Abrishami, 2012)		x			x		
	Budget Constraint Based workflow scheduling (J. Yu & Buyya, 2006a)	x	x					

From a service provider's profitability stance (Table 2.8), several approaches consider the deadline and makespan challenges. Service providers use these challenges as stopping criteria to determine after how long a workflow job must be completed. In most relevant approaches, the deadline is considered to be a principal challenge. Researchers have mentioned that the deadline, budget and makespan are important in the negotiation process between service consumers and the service providers of WfMS in cloud computing. Similarly, SLA is crucial to better manage the execution cost based on consumer decisions. Consequently, these challenges offer "pay-as-you-go" and "on-demand" that directly affect the economics of Cloud WfMS. Both availability and reliability challenges play a crucial

role in improving user satisfaction. This can be done by keeping historical data (spending extra storage cost) for each service request based on the user's input on cost and time, which can be useful in adjusting the QoS parameters for future task scheduling. However, only few approaches are projected to solve these challenges.

2.3.2 System Functionality Challenges

A set of functional system components needs to be designed and developed to meet system functionality requirements. The system functionality of workflow systems can be classified into two major aspects: (i) the basic functional components; and (ii) resource management components (Figure 2.9).

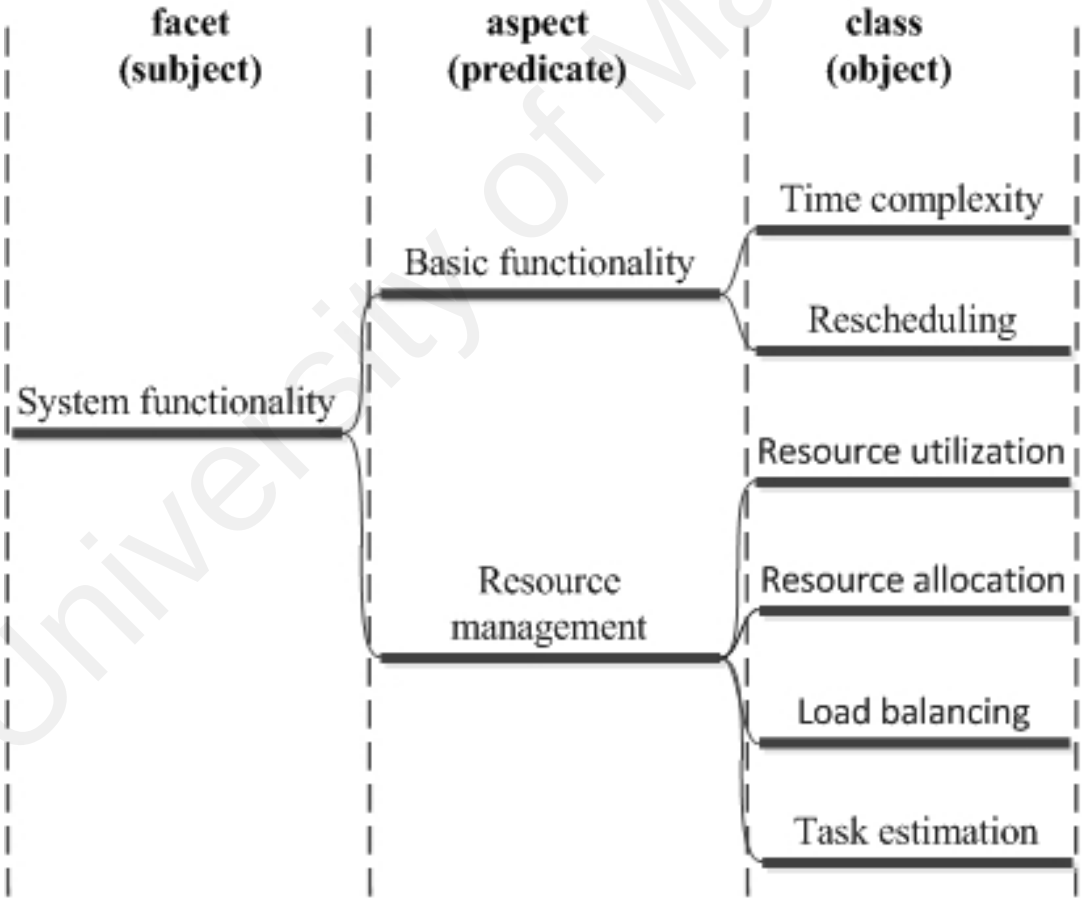

Figure 2.9: Sub-taxonomy of system functionality challenges

2.3.2.1 Types of System Functionality Challenges

The following are two types of the relevant system functionality challenges for SWFS: (i) basic functionality of SWFS challenges; and (ii) resource management challenges.

A-Basic functionality The basic functionality challenges of SWFS can be further divided into two relevant groups: (i) time complexity; and (ii) rescheduling.

Time complexity:

The time complexity of an algorithm can be defined as the time required by the algorithm to run the considered input, which usually represented as the length of iterations. The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms. In this research context, the issue of task mapping on distributed resources commonly falls under the heading of NP-complete problem. For NP-complete problems, there is a correlation between problem size and algorithm execution time. As the problem size increases, the algorithm execution time also increases (Saeid Abrishami, 2013; J. Yu & Buyya, 2006b). Thus, the time complexity of the algorithm needs to be computed to seek out a solution with polynomial time complexity (which may not be optimal)(Abrishami & Naghibzadeh, 2012; Talukder et al., 2009). One of the suitable algorithms is the Genetic Algorithm (GA), which offers a dynamic search technique that requires a quality solution from an enormous search space in polynomial time (Saeid Abrishami, 2013; J. Yu & Buyya, 2006b; S. Kaur & Verma, 2012; Khajemohammadi, Fanian, & Gulliver, 2013; Rodriguez & Buyya, 2017). In addition, the main assumption of calculating the time complexity of SWFS algorithms are:

- The size of the input is very large.
- The worst case scenario is always considered. However, there are several rules that need to be considered while analysing an algorithm (Ambati, Ambati, & Mokhtar, 1991; Tsai et al., 2014):

- There is a need to drop the lower order terms.

- There is a need to drop the constant multiplier.

- The total running time equals the summation of the running times of all the fragmentations.

Traditionally, the estimated running time of heuristic is $O(M \log M)$, where M indicates the number of sub-solutions of each solution of the problem (Tsai et al., 2014; Ambati et al., 1991). The time complexity is different for each type of meta-heuristic (i.e. hybrid meta-heuristic, or hyper meta-heuristic).

Rescheduling: The act of rescheduling is normally applied in the system to adjust particular cloud service dynamics (J. Yu et al., 2005). At times, the service instance may be unavailable or delayed, which causes execution failure of the subsequent service instances to follow through. Thus, the unexpected tasks will delay the user's deadline. For such scenario, rescheduling would be needed to control the violation of contract completion (W.-N. Chen & Zhang, 2009; K. Liu et al., 2010). Additionally, a key notion of the selective rescheduling policy is to assess execution of an individual task and the actual starting time of each task against the estimated starting time to make a proper rescheduling decision (Sakellariou & Zhao, 2004b).

It is impossible to consider every task individually, because rescheduling of tasks to available resources requires more communication time which consequently increases the overall execution cost (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013). Therefore, rescheduling itself places extra workload on the processes of scheduling and execution. This may be associated with the re-evaluation of scheduling cost and the transfer of task cost across resources in conformance with the determined schedule (Sakellariou & Zhao, 2004b).

B-Resources management There are four relevant resource management challenges of

SWFS: (i) resource utilisation, (ii) resource allocation, (iii) Load balancing, and (iv) Task estimation.

Resource utilisation: Several methods consider resource utilisation as one of the performance criteria (Bittencourt & Madeira, 2011; Nargunam & Shajin, 2012). In addition, resource utilisation also deals with the challenges of choosing computing resources in various organisations (Ranaldo & Zimeo, 2009). A pool of processors can be provided to workflow application users' tasks according to the optimal task, time, and resource utilisation method in accordance to the energy level to establish cloud computing performance (Nargunam & Shajin, 2012). In order to achieve cost optimisation, it is essential to offer effective resource utilisation by rendering SWFS application execution more affordable (Bittencourt & Madeira, 2011). Nevertheless, this individual optimisation criterion may pose major limitations that can result in ineffective resource utilisation, unreliable resource use or increasing monetary costs (Prodan & Wieczorek, 2010; Deldari et al., 2017).

Resource allocation: Resource allocation deals with scheduling tasks and the required resources while considering both resource availability and project time. Every resource has three features, namely execution speed, resource ID, and execution cost (Z. Wu et al., 2013).

The resource usage cost is an important factor that acts as an essential determinant in resource selection decisions (Ramakrishnan et al., 2011). Each individual resource usage must be taken into account to measure the direct costs of applications while considering the processing cost of resource allocation (i.e. data storage cost, resource cost, resource computation cost, I/O cost, resource wastage, required network resources, service provider cost, etc.) (Z. Wu et al., 2013, 2010; Q. Tao, Chang, Yi, Gu, & Yu, 2009; Choudhary et al., 2012; Menasce & Casalicchio, 2004). At the same time, the correct resource allocation for

various services has an important function in balancing the QoS (number of processors, duration) (Ramakrishnan et al., 2011; Afzal et al., 2006). It is evident (Ramakrishnan et al., 2011) that there is a high correlation between execution time and overall execution cost. Consider a scheduling situation that demands longer execution time due to the user request of earlier start time. Therefore, it will increase the overall execution cost. As a result, users will be charged with additional resource allocation time in this case even though the resource may be idle (Ramakrishnan et al., 2011).

Load balancing: The physical facilities for providing cloud computing services are distributed over multiple centers. The scheduler needs to allocate the workload to available resources, while dynamically balancing the workload. Consequently, it helps in maximum resource utilisation, which results in improving the overall system performance. However, the scheduler also needs to consider all QoS constraints as requested by the cloud users.

In the literature, two types of load balancing approaches in cloud have been proposed to satisfy the QoS constraints: (i) centralised; and (ii) distributed. Similarly, researchers have categorised the scheduling approaches based on the system information in two classes: (i) static; and (ii) dynamic. In static load balancing, a prior knowledge regarding the requested tasks is required for workload allocation. Conversely, dynamic load balancing does not require any prior knowledge. However, dynamic approaches have high overhead due to the requirement of real time updates regarding system information.

Task estimation: The scheduling approaches are supposed to estimate in advance whether a workflow will be able to meet the requested constraints (e.g., deadline) or not. However, the estimation process may be compromised due to uncertainty in task estimations especially in the case of deadline-sensitive applications (e.g., weather forecasting). Furthermore, the

resource providers find it hard to ensure the resource availability due to the variability and complexity of the underlying resource characteristics and access policies. Researchers have considered three parameters to overcome the aforementioned challenges: (i) Estimated Execution Time (EET); (ii) Estimated data Transfer Time (ETT); and (iii) Estimated Finish Time (EFT). The scheduler needs to include these parameters in the workflow definition to enhance the estimation process by considering the historical results.

2.3.2.2 System Functionality Challenges from WfMS in Cloud Computing

Table 2.9 links cloud system aspects (i.e. system structure and service-driven) with each system functionality of SWFS challenges in cloud computing. In order to increase profitability by meeting system functionality challenges, it is recommended that these challenges must be considered from the service provider's point of view, only. Keeping this recommendation in mind, existing approaches have considered all system functionality challenges as per the service provider's view rather than service consumer.

Table 2.9: Correlation between system functionality challenges and WfMS

	System structure			Service driven		
Challenges	SaaS	PaaS	IaaS	Service consumer	Service provider	Market-oriented
Time complexity	**x**	**x**	**N/A**	**N/A**	**x**	**N/A**
Rescheduling	**x**	**x**	**x**	**N/A**	**x**	**N/A**
Resource utilisation	**N/A**	**x**	**x**	**N/A**	**x**	**N/A**
Resource allocation	**N/A**	**x**	**x**	**N/A**	**x**	**N/A**
Load balancing	**N/A**	**N/A**	**x**	**N/A**	**x**	**N/A**
Task estimation	**N/A**	**N/A**	**x**	**N/A**	**x**	**N/A**

2.3.2.3 System Functionality Challenges from Profitability Aspect

Table 2.10 highlights the results related to service consumer-based approaches and system functionality challenges in SWFS.

Table 2.10: System functionality challenges from the service consumer's profitability view

		System functionality challenges					
Profitability	Model name	Basic functionality		Resources management			
		Time complexity	Rescheduling	Resource utilisation	Load balancing	Resource allocation	Task Estimation
Service customer approaches	CTC (K. Liu et al., 2010)		x				x
	MDP (J. Yu et al., 2005)		x				
	DAG-LOSS & DAG-GAIN (Sakellariou et al., 2007)	x		x			
	ILP (Genez et al., 2012)		x				
	Cost Optimisation and Time Optimisation Scheduling Policies (Salehi & Buyya, 2010)			x			
	Market-oriented hierarchical scheduling strategy (Z. Wu et al., 2013)	x			x	x	
	SC-PCP (Abrishami & Naghibzadeh, 2012)	x	x				
	MODE (Talukder et al., 2009)	x					
	Pegasus framework (Deelman et al., 2005)				x	x	
	Genetic Algorithm (J. Yu & Buyya, 2006b)	x					
	ACS (W.-N. Chen & Zhang, 2009)		x				
	PCP (Saeid Abrishami, 2012)						x
	WORDS (Ramakrishnan et al., 2011)				x		
	MCGP (Tanaka & Tatebe, 2012)						x
	HCOC & MDP (Bittencourt et al., 2012)						x
	Budget Constraint Based workflow scheduling (J. Yu & Buyya, 2006a)	x					

From the service consumer's profitability view (Table 2.10), a large number of approaches (Z. Wu et al., 2013; Abrishami & Naghibzadeh, 2012; Talukder et al., 2009; J. Yu & Buyya,

2006b; Sakellariou et al., 2007; J. Yu & Buyya, 2006a; Tsai & Rodrigues, 2013) measure the time complexity as a basic functionality of WfMS with the intention of determining a solution within polynomial time by using meta-heuristic techniques. Recently, researchers have been focusing on rescheduling challenges to avoid failure of cloud service instances that would cause delays in user-defined deadlines. On the other hand, some models focus on resource management challenges including: (i) resource utilisation (Sakellariou et al., 2007; Salehi & Buyya, 2010); and (ii) resource allocation (Z. Wu et al., 2013; Deelman et al., 2005). The resource management challenges are critical for both service providers and service consumers. For the service provider's point of view, these challenges require the service consumers to manage the rented resources and their workload in order to make efficient usage of resources.

Table 2.11 shows the association between the service provider-related approaches and system functionality challenges in SWFS.

Table 2.11: System functionality of service provider's

		System functionality challenges					
Profitability	Model name	Basic functionality		Resources management			
		Time complexity	Rescheduling	Resource utilisation	Load balancing	Task Estimation	Resource allocation
Service provider approaches	HCOC & MDP (Bittencourt et al., 2012)	**x**			**x**	**x**	
	Time and Cost-constrained scheduling strategy (Ranaldo & Zimeo, 2009)			**x**		**x**	
	RDPSO (Z. Wu et al., 2010)			**x**		**x**	
	(Choudhary et al., 2012)						**x**
	Dynamic Resource Provisioning Techniques (Ostermann et al., 2010)		**x**	**x**		**x**	
	CMSA (Grounds et al., 2009)					**x**	**x**
	Time-Cost trade-off workflow scheduling algorithm (Y. Yuan et al., 2006)	**x**			**x**	**x**	

	CAD (Hsu et al., 2011)						x
	CHPS (Nargunam & Shajin, 2012)						x
	SHEFT (C. Lin & Lu, 2011)				x	x	
	Heuristic designing Scheduling framework (Menasce & Casalicchio, 2004)				x	x	
	Low-Cost Rescheduling Policy (Sakellariou & Zhao, 2004b)	x	x			x	
	HCOC (Bittencourt & Madeira, 2011)		x	x		x	
	WORDS (Ramakrishnan et al., 2011)	x			x	x	
	IC-PCPD2 & IC-PCP (Saeid Abrishami, 2013)	x	x	x		x	
	DCA (Prodan & Wieczorek, 2010)	x		x		x	x
	MINLP (Afzal et al., 2006)	x			x	x	
	Budget Constraint Based workflow scheduling (J. Yu & Buyya, 2006a)	x				x	

From the service provider's profitability stance (Table 2.11), Four approaches (Ostermann et al., 2010; Sakellariou & Zhao, 2004b; Bittencourt et al., 2012; Saeid Abrishami, 2013) are aimed at solving the rescheduling challenge affecting the communication cost of tasks across resources. Similarly, some approaches focus on resource management challenges to provide optimal solutions which can impact the QoS performance of the requested services.

2.3.3 System Architecture Challenges

System architecture design is among the most important challenges in the software development process. The system architecture dictates how the system components are organised and how they interface with each other. Therefore, non-functional requirements are not only influenced by individual system components, but also the system architecture.

Figure 2.10 shows the sub-section presents the system architecture challenges for SWFS.

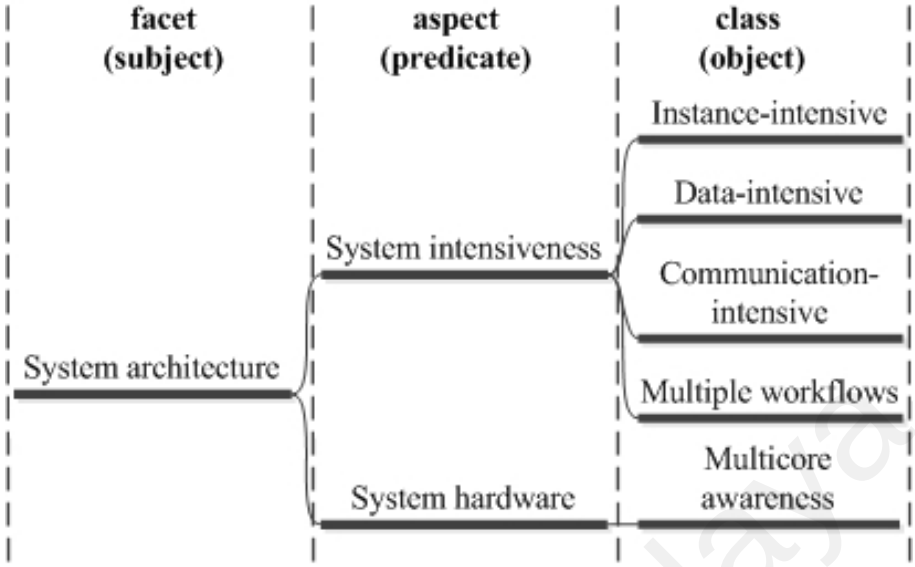

Figure 2.10: Sub-taxonomy of system architecture challenges

2.3.3.1 Types of System Architecture Challenges

The following are two types of the relevant system architecture challenges for SWFS: (i) system intensiveness; and (ii) system hardware.

A-System Intensiveness Workflow application intensiveness affects the system distribution challenges of SWFS in cloud computing environments with different criteria. There are three main types of intensiveness challenges which will be discussed in the following.

Instance-Intensive: Cloud workflow systems need to handle intensive requests for workflow applications, or in other words, WfMS in cloud computing are instance-intensive (Z. Wu et al., 2013; Bittencourt & Madeira, 2013; K. Liu et al., 2010). The execution time can be utilised to determine the estimated time to finish the workflow instance, which would signify how many workflow instances can be completed in a given duration. Thus, a more essential criterion for SWFS is instance-intensiveness compared to the execution time of individual instances (K. Liu et al., 2010). Instance-intensiveness of cloud applications requires new techniques for selecting both data and computation of the distributed resources

so that the execution cost and time of workflow application can be efficiently utilised (J. Yu & Buyya, 2006b). Instance-intensive applications are reflected in several particular e-government and e-business applications.

Data-Intensive: Workflow application is usually defined as a special workflow of the data-intensive type where the quantity and/or size of data are huge. Additionally, a good example of a data-intensive workflow application is Weather Research and Forecasting (WRF). The WRF model requires many stages involving data pre-processing and preparation, actual model simulation, and a post-processing stage. The WfMS in cloud computing should have good scalability to manage various application domains requested like the computation and data-intensiveness of SWFA applications (astrophysics and weather forecasting) (Z. Wu et al., 2013). Thus, data-intensiveness requires high-performance workflow systems in which data-oriented task schedules are a critical issue aimed at maximising the throughput of input and output in a particular workflow (Tanaka & Tatebe, 2012; Horiuchi & Taura, 2012). Every step may require intensive computation and/or include a large quantity of data processing and transfer (Z. Wu et al., 2010; Q. Wu et al., 2013). However, conducting successful scheduling of data-intensive applications in cloud computing environments is normally a challenging task (Srinivasan, 2012; Szabo et al., 2014). Many of the cloud applications are data-intensive with large datasets, resulting in a large number of workflow instances, which leads to a high requirement for tasks to be optimally scheduled within the cloud's network devices (J. Yu et al., 2005; Sudha & Monica, 2012). Owing to this fact, data transfer from one computed node to another requires longer execution time (Pandey et al., 2010). Nevertheless, SWFAs normally entail processing a large quantity of data and activities that are computationally intensive. Therefore, greater storage and communication cost is required compared to the computation and execution cost of these data. The execution cost of a large SWFA dataset normally consumes higher cost compared

with a smaller SWFA dataset, this is due to the reason that the large SWFA dataset requires longer execution time to complete the submitted SWFA tasks, which ultimately required higher or more powerful computational resources (Ramakrishnan et al., 2011; Z. Wu et al., 2010).

Communication-Intensive: Some models are considered communication-intensive applications, which are unlike the instance-intensive applications. However, workflows that are considered communication-intensive in cloud computing need a high amount of computation on servers but may sometimes cause failures (Y. Yang et al., 2008; Stephanakis et al., 2013; Koseoglu & Karasan, 2010). The Communication to Computation Ratio (CCR) is calculated by determining the average communication cost divided by the computation cost average in the target system (Delavar & Aryan, 2012; Amandeep Verma, 2012). The variation between multiple workflows and communication-intensive workflows is that the latter are multiple instances of one workflow, while multiple workflows are an integration of various workflows (Tanaka & Tatebe, 2012; Bittencourt & Madeira, 2013).

Multiple workflows: Traditionally, the WFS process is performed for a single workflow. After the emergence of cloud and grid computing paradigms, researchers have focused on multiple workflows scheduling. The multiple workflows remain a crucial challenge since it requires providing several services to multiple users at the same time. Furthermore, the service provider has to consider multiple QoS constraints for each user. It is important to differentiate between communication intensive and multiple workflows for better understanding. The communication intensive workflows are multiple instances of the same workflow, while multiple workflows are various types of workflows. In other words, communication intensive workflows have the same structure and multiple workflows may have completely different structures.

B-System hardware There is only one system hardware challenge with WfMS, that is,

multicore awareness.

Multicore awareness: Multicore awareness can be described as the existence of a computer system that provides super processing power via multiple cores. Such systems must be given special attention when developing the scheduling algorithms for resource selection (Bittencourt et al., 2012). Similarly, workflows are submitted by many clients to a scheduler who assigns the requests to a memory-managed multicore machine in a cloud for execution purposes (Nargunam & Shajin, 2012). Furthermore, every multicore machine in the platform can perform requests from multiple services concurrently, as the machine executes each request as an independent thread. The cost of hardware and surge of storage capacity and computing power, together with the emergence of the multicore architectural layout and modern super-computers comprising hundreds of thousands of cores, are some of the key components contributing to the surging interest in cloud computing (Foster et al., 2008; Byun et al., 2011). The multicore characteristic besides cost awareness can provide makespan as low in cost as the user demands. As such, the user can control the cost by manipulating the desired time for workflow execution based on preference (Bittencourt & Madeira, 2011). Utilising multicore resources offers the benefit of reducing communication cost as well as a way to minimise the makespan.

2.3.3.2 System Architecture Challenges from WfMS in Cloud Computing

Table 2.12 associates WfMS in cloud computing (i.e. system structure and service driven) on each of the system architecture of SWFS challenges in cloud computing. It can be concluded that the system architecture for WfMS in cloud computing should follow the general architecture of cloud software, but at the same time, it also needs to be adapted according to different system requirements.

Table 2.12: Correlation between system architecture challenges and WfMS

	Distributed system structure			Service driven		
Challenges	SaaS	PaaS	IaaS	Service consumer	Service provider	Market-oriented
Instance-intensive	**x**	**x**	**N/A**	**N/A**	**x**	**N/A**
Communication-intensive	**x**	**x**	**x**	**N/A**	**x**	**N/A**
Data-intensive	**x**	**x**	**N/A**	**N/A**	**x**	**N/A**
Multiple workflows	**x**	**x**	**N/A**	**N/A**	**x**	**N/A**
Multicore awareness	**x**	**x**	**x**	**N/A**	**x**	**N/A**

2.3.3.3 System Architecture Challenges from Profitability Aspect

Table 2.13 depicts various service consumer-related approaches and targeted system architecture challenges in SWFS.

Table 2.13: System architecture challenges from a service consumer perspective

		System architecture challenges				
Profitability	Model name	Intensiveness				System hardware
		Instance-intensive	Data-intensive	Communication-intensive	Multiple workflows	Multicore awareness
Service consumer	CTC (K. Liu et al., 2010)	**x**				
	Transaction Intensive Cost-constraint Algorithm (Y. Yang et al., 2008)	**x**		**x**	**x**	
	(Choudhary et al., 2012)			**x**		
	ILP (Genez et al., 2012)			**x**		**x**
	Market-oriented hierarchical scheduling strategy (Z. Wu et al., 2013)	**x**	**x**		**x**	
	BGQoS (Albodour et al., 2012)	**x**	**x**			
	ACS (W.-N. Chen & Zhang, 2009)		**x**			

From a service consumer's profitability perspective (Table 2.13), the intensiveness-based challenges have been applied for different components of WfMS in cloud computing architecture. The instance-intensive, data-intensive and communication-intensive challenges significantly affect the system distribution of workflow application while considering the service consumer's requirements. For better resource utilisation, four approaches

(Z. Wu et al., 2013; K. Liu et al., 2010; Y. Yang et al., 2008; Albodour et al., 2012) focus on instance-intensive challenge while reducing the execution cost and time of workflow application. On the other hand, some approaches (Z. Wu et al., 2013; Y. Yang et al., 2008; Genez et al., 2012; W.-N. Chen & Zhang, 2009) consider communication and data-intensiveness in workflow application to reduce the complexity of calculating the required, extensive computational-in WfMS in cloud computing. However, only one approach (Genez et al., 2012) considers the multicore awareness challenge of system hardware. Multicore awareness challenge is an important concern for both service provider and service consumer. Usually both service consumers and service providers are interested to know the cost of rented multicore VMs. Therefore, future work should focus on proposing an approach that efficiently solves multicore awareness challenge.

Table 2.14 shows the association results between the service provider-based approaches and targeted system architecture challenges in SWFS.

Table 2.14: System architecture challenges from a service provider perspective

		System architecture challenges				
Profitability	Model name	Intensiveness				System hardware
		Instance-intensive	Data-intensive	Communication-intensive	Multiple workflows	Multicore awareness
Service provider	CMSA (Grounds et al., 2009)					**x**
	Time and Cost-constrained scheduling strategy (Ranaldo & Zimeo, 2009)		**x**			
	PSO (Pandey et al., 2010)		**x**			
	CPPS (Q. Wu et al., 2013)					**x**
	SHEFT (C. Lin & Lu, 2011)		**x**			
	Heuristic designing Scheduling framework (Menasce & Casalicchio, 2004)		**x**			
	HCOC (Bittencourt & Madeira, 2011)					**x**
	MCGP (Tanaka & Tatebe, 2012)		**x**			

	ACO (H. Liu et al., 2011)			**x**		
	MQMW (Xu et al., 2009)				**x**	

From a service provider's profitability view (Table 2.14), it is evident that the intensiveness-related challenges play an important role in the system architecture of cloud workflow. Thus, several approaches focusing on data-intensiveness for each system in WfMS is required for large magnitude data computation and communication. However, none of the service providers' approaches measure the instance intensiveness of workflow applications. One possible reason behind this deficiency is that this challenge is closer to the service consumer than the service provider. Nonetheless, one approach (H. Liu et al., 2011) considers communication intensiveness, whereas three approaches (Grounds et al., 2009; Bittencourt & Madeira, 2011; Q. Wu et al., 2013) focus on the multicore awareness challenge.

2.3.4 WfMS in Cloud Computing

This section classifies the key aspects of WfMS in cloud computing (i.e. cloud system model, cloud service driven). These aspects are created based on the workflow reference model proposed by the Workflow Management Coalition (WfMC) (Coalition, 2005). The cloud system model contains three levels (Coalition, 2005; Ramakrishnan et al., 2011; Abrishami & Naghibzadeh, 2012; Ranaldo & Zimeo, 2009; Z. Wu et al., 2010; Genez et al., 2012) as follows:

(i) workflow application level that represents the development and management known in cloud model as Software as a Service (SaaS): The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or

control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

(ii) middleware level that represents the workflow management enactment service known in cloud model as Platform as a Service (PaaS): The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

(iii) Infrastructure level that represents the unified resources known in cloud model as Infrastructure as a Service (IaaS): The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

On the other hand, cloud driven services play a vital role in the WfMS focusing on three main aspects including: service consumer, service provider, and market-oriented (Z. Wu et al., 2013; Salehi & Buyya, 2010; W.-N. Chen & Zhang, 2009; Y. Yuan et al., 2006; Malawski et al., 2012).

(i) Service consumer: By referring to NIST (SP 500-292.) definition, the cloud service consumer can be defined as a principal stakeholder that uses the cloud computing services (i.e. Software, Platform or Infrastructure as a Service). Thus, a cloud consumer represents a person or organisation that maintains a business relationship with, and uses the service

from, a cloud service provider. A cloud consumer plays an important role by browsing the service catalog from a cloud provider, requests the appropriate service, sets up service contracts with the cloud provider, and uses the service. The cloud consumer may be billed for the service provisioned, and needs to arrange payments accordingly. In simple terms, the consumer refers to an entity that utilises any given service to perform an action and receive a result in return. For example, a customer opens a bank account. In this case, the entity is the "customer", the action is "open", and the result is the "bank account".

(ii) Service provider: By referring to NIST (SP 500-292) definition, the cloud service provider can be defined as a company that offers some component of cloud computing – typically Infrastructure as a Service (IaaS), Software as a Service (SaaS) or Platform as a Service (PaaS) – to other businesses or individuals. The service provider offers several services with different QoS (e.g., CPU type and memory size) and different prices for each task of every workflow based on business service contracts (Z. Wu et al., 2013; Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013; Verma & Kaushal, 2017). The service provider charges the service consumers for using the service based on the amount of allocated volume, and possibly for the number of I/O transactions from/to outside the Cloud (Abrishami & Naghibzadeh, 2012).

(iii) Market-oriented: Market-oriented is one of the most important aspects which differentiate a WfMS in cloud computing from its other counterparts in the business model (Z. Wu et al., 2013). With the market-oriented business model in cloud computing environments, resource managers should be able to meet the user specified QoS requirements (mainly on makespan and cost) for individual workflow instances as well as minimise the overall makespan and cost for multiple workflow instances running concurrently in WfMS (Z. Wu et al., 2013; X. Liu, 2011).

2.3.5 Discussion on Cost Optimisation Challenges

Cost optimisation challenges of SWFS have been affected by different workflow system structure and service-driven aspects. Therefore, the devised taxonomy considers current state-of-the-art challenges and SWFS profitability in cloud computing based on QoS performance aspects, system functionality as well as system architecture.

From analysing the QoS performance challenges, it can be concluded that with respect to the issues of workflow execution optimisation, the relevant determinants such as monetary cost, temporal cost, security, and reliability need to be optimised (J. Yu & Buyya, 2006b; Garg et al., 2010). Therefore, the overall performance can be assessed in several ways, but mainly by the scheduling success rate. Similarly, the ability to satisfy the QoS of many users is also a good indicator of determining system performance (Xu et al., 2009). From the QoS constraints, scheduling should adjust to the workflow models' constraints, such as cost constraints of the utility workflow process model or time constraints for the timed workflow process model, or both. In the context of QoS performance challenges, researchers have mentioned that the deadline, budget and makespan are important for the negotiation process between the service consumers and service providers of WfMS in cloud computing. Thus, a number of approaches focus on budget, deadline and makespan, mainly due to the adaptability of these challenges in Service Level Agreement (SLA) and market-oriented aspects while determining the required cloud service speed. One of the widespread system performance measures is the deadline, which has a significant effect on algorithm performance.

There is a direct relationship between deadline and cost (i.e. the cost would be higher for a fixed deadline). Thus, in order to examine cost performance with deadline-conscious SWFS algorithms, users should reduce the cost of workflow execution. Consequently, the workflow would be completed before the user's stipulated deadline. Similarly, these

challenges experienced by service provider profitability act as a stopping criterion to determine how long the workflow job must be completed. On the other hand, both availability and reliability challenges play a crucial role in improving user satisfaction. This is done by keeping the historical data for each service request based on the user's input on cost and time, which can be useful to adjust the QoS constraints for future task scheduling.

Figure 2.11 depicts the frequency of QoS challenges reported in the literature. From Figure 2.11, it can be observed that Make-span challenge has attained major attention (31%) of researchers compared to other challenges in this category. This is due to the fact that these challenges are related to execution cost and execution time of SWFS, which is highly considered by the service providers and demanded by the service consumers. In contrast, less attention has been given to the other cost related challenges (i.n. security, availability, reliability and SLA). Therefore, important execution cost and execution time related QoS challenges have been focused.

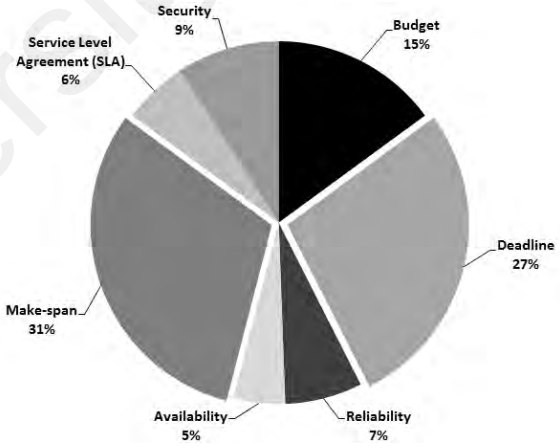

Figure 2.11: The frequency of the QoS performance challenges

As a recommendation for the reported QoS performance challenges, the multi-criteria based cost optimisation should be focused on providing optimal solutions to SWFS problems (Yassa, Chelouah, Kadima, & Granado, 2013; B. Liu, Wu, Xu, Hu, & Cheng,

2014; Grandinetti et al., 2013; Szabo & Kroeger, 2012). Also, researchers ought to give more attention to hybrid approaches by combining the strengths of existing approaches to provide more cost-effective solutions to SWFS problems. Since current state-of-the-art approaches lack in considering reliability, availability, and response time as QoS performance challenges, considering such quality attributes may have a positive impact on the overall quality of the proposed approaches.

From analysing the system functionality challenges, it can be concluded that a large number of approaches (J. Yu et al., 2005; Z. Wu et al., 2013; Abrishami & Naghibzadeh, 2012; Talukder et al., 2009; J. Yu & Buyya, 2006b; Sakellariou et al., 2007; J. Yu & Buyya, 2006a) measure time complexity, which requires finding a high quality solution in polynomial time from the available large search space. Also, researchers should concentrate on rescheduling challenges that affect the communication cost of tasks across resources to avoid failure of cloud service instances. On the other hand, some models focus on resource management challenges to provide optimal solutions which can negatively impact the QoS performance of the requested services. Resource utilisation greatly affects the cost profitability which is a major concern for service providers. In the literature, it has been found that multiple workflows pose a very important challenge. This type of workflow has multiple possible instances between the workflows (Sharif et al., 2013; K. Liu et al., 2010). The multiple workflows may have completely different structures while transaction-intensive workflows have the same structure. However, only few scheduling approaches have been developed for multiple workflows, which indicates more studies in this area should be undertaken.

Figure 2.12 presents the frequency of system functionality challenges. The researcher found that data-intensive challenge has attained major attention (37%) of researchers compared to other challenges in this category. Similarly, multicore-awareness also

achieved high attention (21%) compared to other system functionality challenges.

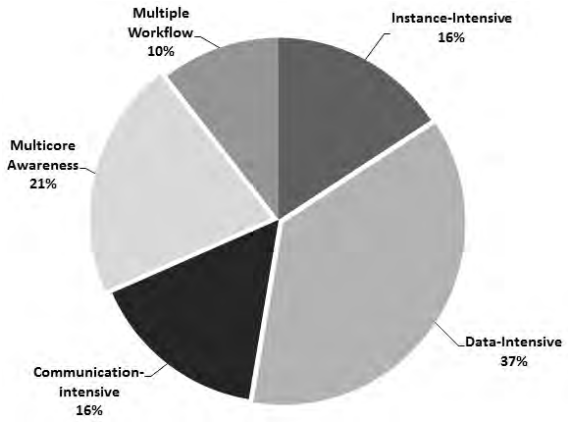

Figure 2.12: The frequency of the system functionality challenges

As a recommendation to the mentioned system functionality challenges, service-oriented computing should be aimed at designing applications by adopting model-driven approaches, and in particular, those using time-based composition of distributed and loosely coupled services (e.g., Service Oriented Architecture) (Ranaldo & Zimeo, 2009; L. Zhao et al., 2012). Therefore, service-oriented computing should be at the core of future research since it is dynamically acquired and managed in SWFS.

By analysing the system architecture challenges, it is evident that the intensiveness-based challenges affect workflow application system distribution while reducing execution cost and time. Similarly, intensiveness-related challenges play an important role in the system architecture of cloud workflow. Several approaches focus on data-intensiveness for each system in WfMS. This requires large quantities of data computation and communication. However, the SWFS in cloud computing is currently facing a major challenge in handling data-intensive applications, especially the data locality of application (Foster et al., 2008). On the other hand, none of the service providers' approaches take into consideration the instance-intensiveness of workflow application. One possible reason is that this challenge is closer to the service consumer than the service provider. Similarly the multi-processor

and multicore scheduling on a modern system has attracted an increasing amount of consideration (Q. Wu et al., 2013).

Figure 2.13 shows the frequency of system architecture challenges as reported in the literature. The time-complexity remains a key focus (28%) of researchers in this class of SWFS challenges. In contrast, some work (18%) has focused on solving rescheduling challenge.

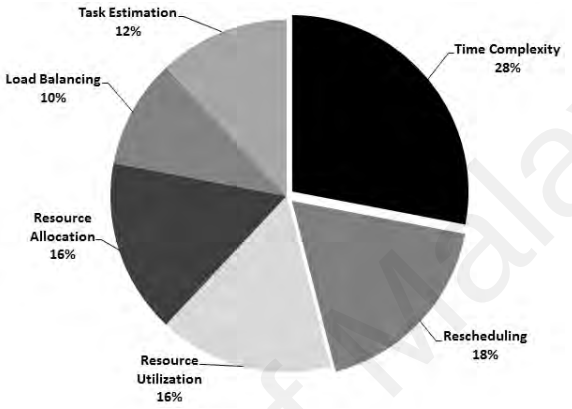

Figure 2.13: The frequency of the system architecture challenges

As a recommendation to the mentioned system architecture challenges, Due to the scale of workflow application (i.e. social application) expansion, requiring long SWFS time to process, it is suggested to have more optimised solutions to handle this massive growth (Z. Wu et al., 2013). Then again, SWFS systems must be given special attention when developing scheduling algorithms for resource selection (Bittencourt et al., 2012). With advances in multicore technology in the future, there are possibilities of strong dependencies on the cost and performance of Infrastructure as a Service (IaaS), especially the computing resources (Foster et al., 2008). However, only few approaches reflect on the multicore awareness challenge of system hardware. This is mainly due to the fact that system hardware specifications are not a concern of the service consumers.

2.4 Cost Optimisation Approaches of SWFS

This section reviews relevant cost optimisation Scientific Workflow Scheduling (SWFS) approaches in cloud computing. Table 2.15 presents the existing approaches, strengths of the underlying optimisation, and limitations for all considered approaches.

Table 2.15: Cost optimisation SWFS approaches

Approach	Strength of underlying optimisation	Limitation
Market-oriented hierarchical scheduling strategy (Z. Wu et al., 2013))	This strategy offers provisional exploration in market-based cloud workflow structures. It assures suitable QoS requirements are imposed by users while decreasing the overall running cost of the workflow system.	Provisional exploration is offered only in market-based cloud workflow structures. There is a need to consider utility structure.
SaaS Cloud Partial Critical Paths (SCPCP) (Abrishami & Naghibzadeh, 2012))	The cost of workflow execution is reduced while the user-determined deadline for the Software-as-a-Service (SaaS) framework is met. Schedules workflow based on QoS using a PCP.	The IaaS and pricing for cloud processing frameworks are not supported by this technique and data transaction cost is unaffected.
Workflow Orchestrator for Distributed Systems Architecture (Ramakrishnan et al., 2011))	Extracts the disparities and encapsulates the QoS features offered by the cloud structure. Allows competent organisations at moderate cost for batch queue with or without public resource management.	The authors found that this model is deficient in enhancing sequential resources possibly requiring higher cost.
Compatibility of Hybrid Processor Scheduler (Nargunam & Shajin, 2012))	Execution cost is lower since resources are chosen based on their energy levels and various forms of applications concerning high processing and storage space demands are characterised.	This method is only applicable for hybrid systems and is based on resources. However, other performance features, such as throughput, are not considered.
Critical Path-based Priority Scheduling (CPPS) (Q. Wu et al., 2013))	The approach outperforms the traditional fair-share scheduling policy commonly adopted in real systems. Deployed and executed a test bed network by adopting an on-node scheduling policy, while improving the workflow performance of the mapping scheme.	It was found that the CPPS model fails to achieve optimised interaction between task mapping and the scheduling scheme.
Dynamic resource provisioning techniques (Ostermann et al., 2010))	Achieved a cost-effective execution of SWFAs by implementing a just-in-time algorithm. This technique uses a combination of cloud resources via dynamic provisioning of cloud resources when cloud resources are unavailable.	Each task is scheduled when it becomes ready by implementing a just-in-time algorithm.
Ant Colony Optimisation (ACO) (H. Liu et al., 2011))	Improved cloud service performance in terms of reliability, response time, cost, and security. ACO utilises a default rate to explain the ratio of cloud system service providers that break the SLAs of the SWFS.	The authors did not consider the effect of QoS constraints in SLA from the users' perspective.

Continued on next page

Table 2.15 – continued from previous page

Approach	Strength of underlying optimisation	Limitation
Genetic Algorithmic (GA) (J. Yu & Buyya, 2006b))	Monetary cost is reduced, and at the same time, user budget constraints and execution time are reduced simultaneous with meeting the user deadline constraint. Provided a dynamic search method by offering high-quality solutions for a vast search area in polynomial time by using the evolutionary principle.	The authors found that GA requires a longer running time than other heuristic techniques.
Budget constraint based workflow scheduling (J. Yu & Buyya, 2006a))	Presented a cost-based scheduling heuristic to minimise execution cost and time while meeting the user's budget. This method adopts GA to minimise workflow execution cost within a certain deadline.	It was observed that this model supports only specific types of services and does not consider duplication of critical tasks to meet users' QoS requirements.
Transaction intensive cost-constraint algorithm (Y. Yang et al., 2008))	Aimed at minimising cost while meeting user-determined deadlines. This algorithm offers a graph of just-in-time cost relations during workflow execution by utilising intensive transaction settings as well as considering the specified budget.	The authors mentioned that the service provider's performance is not considered in this algorithm.
Multi-cost job routing and scheduling (Stevens et al., 2009))	An algorithm for polynomial complexity is provided. Resource reservations are enhanced and the start time for data transmission is identified as well as for task completion.	The communication and computational parameters of monetary cost are not completely considered.
Integer Linear Programming (ILP) technique (Genez et al., 2012))	ILP utilises two heuristic methods that are competent when deadlines are substantial. Reduced the monetary cost while addressing the deadline determined by the cloud users in the SLA contract.	For manifold workflow scheduling, the authors mentioned that this technique does not consider the mechanism's fault tolerance in a similar group of resources.
Multiple QoS of Multi-Workflows (MQMW) (Xu et al., 2009))	This model is used to minimise execution cost while prominently enhancing scheduling success rates. Scheduling success rates are prominently enhanced. It is designed for several workflows with various QoS requirements.	According to the authors, MQMW is not applicable for other QoS constraints such as execution time, availability, etc.
Scalable-Heterogeneous-Earliest-Finish-Time algorithm (C. Lin & Lu, 2011))	Execution time optimisation is achieved while elastic runtime scalability is attained. The authors mentioned that SHEFT is more flexible than other SWFS models as it efficiently schedules on-demand size of a workflow along with allocated resources.	Apparently this algorithm is deficient in indicating accurate runtime prediction, which acts as a prerequisite of scheduling.
Particle Swarm Optimisation (PSO) (Pandey et al., 2010))	It is for designing a heuristic that utilises PSO by formulating a framework for task-resource mapping to reduce the overall completion cost. PSO incurs large amounts of data and execution cost.	The authors mentioned that the proposed algorithm ignores the temporal cost of service providers.

Continued on next page

Table 2.15 – continued from previous page

Approach	Strength of underlying optimisation	Limitation
Multi-Constraint Graph Partitioning (MCGP) (Tanaka & Tatebe, 2012))	Reduced the cost of communication while minimising the workflow execution time. MCGP utilises the partitioning DAG graph that is applied for a complicated Cloud workflow. The time elapsed for file size access from remote nodes is decreased as well.	The MCGP method does not consider the input/output files of each job, which ultimately affect the dynamic changes of workflow applications.
Hybrid Cloud Optimised Cost model (HCOC) (Bittencourt & Madeira, 2011))	Improved SWFS by making use of multicore resources and minimising the monetary cost within a deadline while offering affordable makespans to users. HCOC imparts sufficient processing power by determining which resource ought to be leased from the public cloud.	The authors mentioned that this model does not consider the workflow execution time required by a user in a single-level SLA contract.
Workflow-aware Preprocessing Provisioning Dynamic Scheduling (WPPDS) (Shi, 2014))	An elastic resource provisioning and task scheduling mechanism have been proposed to perform SWFAs and submitted at unpredictable time in cloud. The aim is to finish as many high-priority workflows as possible by considering the budget and deadline aspects. The evaluation of this mechanism shows stable performance with inaccurate parameters and task failure.	The approach should be tested within a real SWFA environment by considering the data communication cost and storage cost for executing larger workflows in cloud.
SciCumulus (Viana et al., 2011))	A parallel technique to define the number and types of VMs and to design the parallel execution strategy for a SWFA. Model a cost based approach to satisfy the QoS and help determine an adequate configuration of the environment according to restrictions imposed by service consumers.	The approach is lacking in providing a comprehensive evaluation environment in order to test the total execution cost and execution time.
Dynamic Provisioning Dynamic Scheduling (Malawski et al., 2012))	A series of adaptive scheduling algorithms for SWFAs cloud-based have been proposed to provide a dynamic dimensioning and vertical scaling during workflow execution to comply with service consumers' constraints. This work represents good performance results by bringing opportunities for modifying the number of VMs.	Lack in considering the optimisation of the initial virtual machine allocation. Also, this approach is unable to handle unpredictable workloads.

Continued on next page

Table 2.15 – continued from previous page

Approach	Strength of underlying optimisation	Limitation
Cost-Effective Virtual Machine Allocation Algorithm within Execution Time Bound (Zhu et al., 2012))	A two-step heuristic workflow mapping (scheduler) has been used to maximise the resource utilisation while minimising the overhead of the cloud. The delay of workflow execution has also been considered to calculate makespan and guarantee the user required deadline, to reduce the overhead of start-up and shutdown of a virtual machine.	The authors have taken the VM launching's overhead variation as a key variable for designing resource allocation algorithms only. However, there are several other constraints which should also be considered (e.g., the predication of workload, QoS requested by users)
Critical-Greedy (CG) (X. Lin & Wu, 2013))	An algorithm (MED-CC) has been designed for workflow scheduling problem to minimise end-to-end delay while meeting the budget constraint. Furthermore, the authors have proposed a Critical-Greedy algorithm using GAIN approach that only imposes rescheduling process on dynamic critical tasks.	The reported experimental results prove the performance benefits of the approach but not completely achieve the required cost optimisation.
SO-Routine and MO-Routine algorithm (Szabo et al., 2014))	Evolutionary approach based algorithm for multi-objective optimisation has been proposed as a solution to scheduling of SWF applications. Also, the authors designed a novel representation of two independent chromosomes, one for mapping and one for ordering.	This method is more time consuming than other heuristic based approaches. The authors used only two Amazon EC2-based clouds which are not enough for evaluating a multi-cloud challenge.

2.4.1 Existing Meta-heuristic Approaches for Cost Optimisation of SWFS

In the literature, there are several types of meta-heuristic combinations that have been proposed for the scheduling problem. The most efficient technique is by hybridising two or more algorithms into a single meta-heuristic algorithm (Figure 2.14). The hybrid algorithms have shown a good performance for cost optimisation of SWFS problem by leveraging strengths of the combined algorithms. The single meta-heuristic algorithms are easy to implement in WfMS of cloud environment compared with hybrid meta-heuristic algorithms; however, single meta-heuristic algorithms lack in obtaining optimal or satisfactory results (Tsai et al., 2014). This is due to the fact that at each iteration of the convergence process, single meta-heuristic algorithms take a longer execution time to schedule the submitted SWFAs. Therefore, researchers have started to introduce the

hyper-heuristic algorithm to find a more effective solution that can use "one and only one" meta-heuristic algorithm at each iteration. The basic idea of the traditional hyper meta-heuristic algorithm is to leverage and integrate the strengths of employed meta-heuristic algorithms, by integrating them into a single algorithm (Tsai et al., n.d., 2014). Hyper meta-heuristic proposed two levels of heuristic including High Level Heuristic (HLH) and Low Level Heuristic (LLH). At each iteration, the selection operator of HLH is used to choose one candidate meta-heuristic algorithm to be a scheduling algorithm from the LLH pool of meta-heuristics. In this way, the hyper meta-heuristic algorithms can then provide a high search diversity to increase the chance of finding more optimal solutions at later iterations while minimizing the execution time.

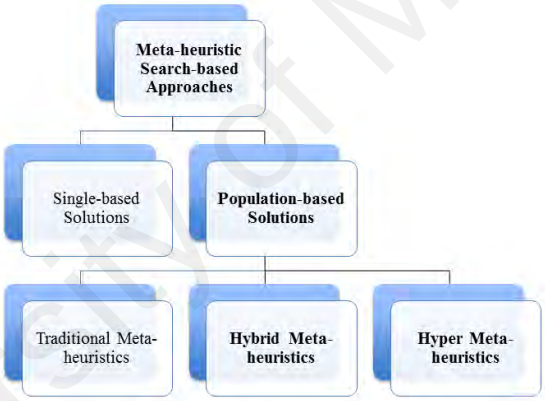

Figure 2.14: Existing meta-heuristic approaches

2.5 Summary

Scientific Workflow Scheduling (SWFS) remained an active area of research since emergence of cloud computing for SWFAs. One of the major aims of SWFS is to reduce the completion time as well as the total computional cost of SWFS in cloud computing especially for complex tasks like SWFA. The execution time must be taken into consideration while scheduling SWFA tasks into the available computational resources. Therefore, from the state-of-the-art cost optimisation of SWFS, it has been observed

that several concepts have been considered to optimally schedule the SWFA tasks to the available computational resources. In this chapter, the current aspects, parameters, challenges and approaches of cost optimisation for SWFS in cloud computing environment has been extensively reviewed.

From the detailed analyses, it has been observed that majority of cost optimisation of SWFS approaches considered various aspects of SWFS: heuristic methods (57%), directed acyclic graph (85%), dynamic technique (77%), and predicted workload (62%). Regarding the QoS constraints, there is a variety of research focus such as privacy, response time, reliability, and security. From the parameters aspect, most of the proposed approaches have applied both temporal and monetary cost parameters during various stages of SWFS. Researchers have paid more attention to profitability from the point of view of service providers than the service consumers. Based on the extensive literature review, the cost optimisation challenges of SWFS (i.e. QoS, system functionality, and system architecture) has been identified. Moreover, to highlight the underlying association, the correlation between cost-optimisation of SWFS challenges and other aspects of SWFS (i.e. WfMS in cloud computing and cost-aware profitability) has been identified.

Several opportunities and recommendations have been suggested to assist in developing an approach for cost optimisation. In the context of identified cost optimisation challenges, some aspects got less attention than others of similar category. Similarly, designing a hybrid and hyper approaches (by integrating the features of existing meta-heuristic search-based algorithms) would certainly improve the scalability challenge due to concurrent processing. Furthermore, it is evident from the observed trends related to SWFS that researchers are mainly focusing on large-scale data-intensive applications for evaluation purpose. Therefore, future research should focus on proposing more optimised approaches by considering the strengths of heuristics and meta-heuristic methods.

CHAPTER 3: RESEARCH METHODOLOGY

This chapter discusses the research methodology that has been adopted for this research. Experts from academia have also been consulted to validate the devised research methodology in order to meet the main objective of this research. Note that the key objective is to propose an approach that can find an optimal solution by addressing the cost optimisation challenge of SWFS. The proposed approach ultimately helps in meeting the user's requirements in executing the SWFA. To achieve the formulated research objectives, three main stages have been followed: (i) formulation stage, (ii) approach development stage, and (iii) evaluation and analysis stage. The first stage (i.e. formulation stage) involves an in-depth analysis of different cost optimisation perspectives of SWFS based on the synthesis of the conducted extensive literature review (described in Chapter 2). Note that the conducted literature review reports relevant SWFS challenges, aspects, parameters, and approaches.The second stage (i.e. approach development stage) defines the main considered activities of the proposed time completion driven hyper-heuristic approach.

The proposed approach contains two main parts, the cost-optimisation model of SWFS and the proposed dynamic hyper-heuristic algorithm. The cost-optimisation model helps to understand the mapping and scheduling processes of workflow tasks by considering the scheduling stages along-with completion time and total computational cost parameters. While, the proposed hyper-heuristic algorithm is considered as a new advanced technique that is capable of accelerating the run-time of the meta-heuristic algorithm by employing four well-known population-based meta-heuristic algorithms, which act as Low Level Heuristic (LLH) algorithms. The employed algorithms are: (i) Genetic Algorithm (GA), (ii)Particle Swarm Optimisation (PSO), (iii) Invasive Weed Optimisation (IWO), and (iv) Hybrid Invasive Weed Optimisation (HIWO). In addition, the proposed dynamic hyper-

heuristic algorithm enhances the native random selection way of existing hyper-heuristic algorithms by incorporating the best computed workflow completion time, which acts as a high-level selector to pick a suitable algorithm from the pool of LLH algorithms after each run. The main aim of the proposed completion time driven hyper-heuristic approach is to reduce the completion time, and total computational cost to execute the SWFA. At the last stage (i.e. evaluation and analysis stage), the research methodology adopted to evaluate the proposed approach based on the guidelines of state-of-the-art cost optimisation SWFS approaches have been discussed. Figure 3.1 provides a high-level view of the considered stages including a number of activities to achieve each of the formulated research objectives.

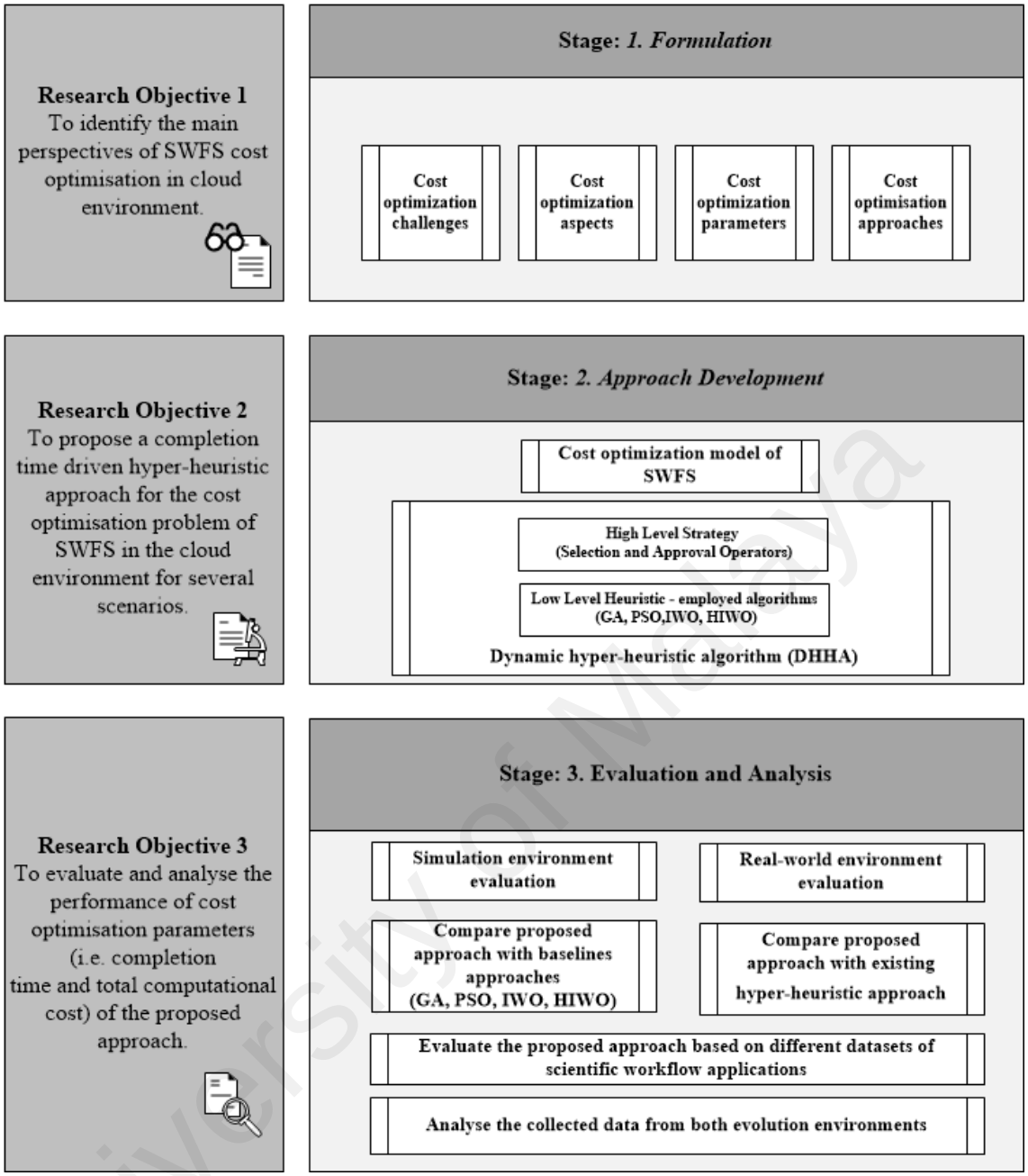

Figure 3.1: High-level view of research methodology stages

The following sections provide detailed explanations for each of the defined stage of adopted research methodology.

3.1 Formulation Stage

This section focuses on the main observations concluded from the conducted extensive literature review as described in Chapter 2 that aims in achieving the first objective of

this research. The classification and discussion of cost optimisation aspects, parameters, challenges and approaches of SWFS help in providing comprehensive guidelines, and recommendations for proposing a cost optimised SWFS approach (Figure 3.2).

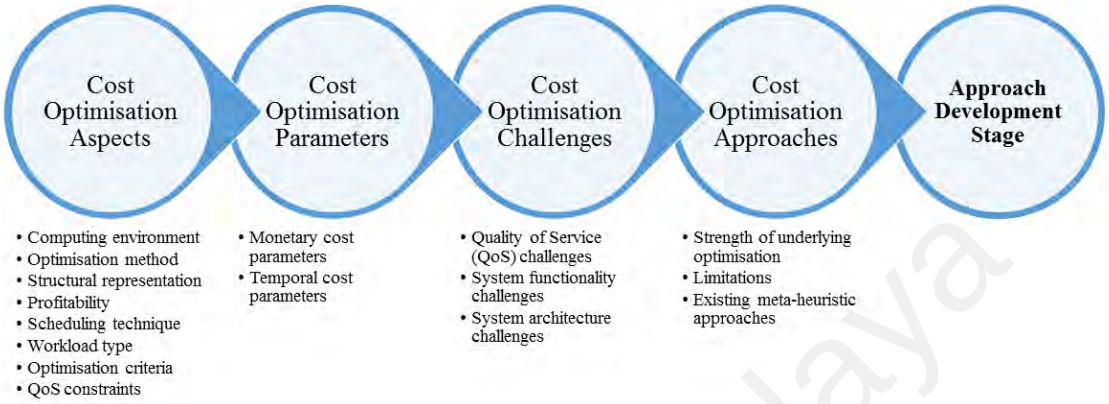

Figure 3.2: The activity of formulation stage

3.1.1 Cost Optimisation Aspects of SWFS

From the cost optimisation aspects' point of view, Figure 3.3 llustrates the main options selected from analysing the specification of computing environment, optimisation method, structural representation, scheduling technique, workload type, and QoS constraints. The details about these aspects have been provided in Section 2.1.

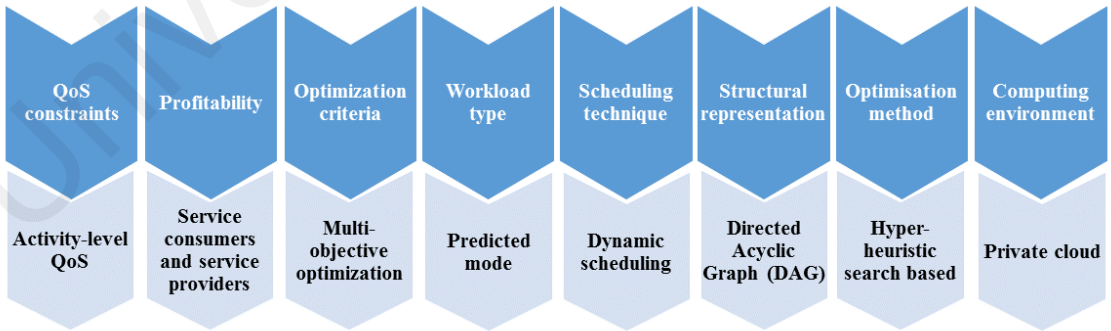

Figure 3.3: Cost optimisation aspects

Computing environment: A large number of cost optimisation SWFS approaches are implemented using *private cloud*. The *Private cloud* does not require any remote computa-

tional infrastructure (Infrastructure as a Service) where the requested workflow tasks are being processed. Therefore, the advantages of private cloud model for experimentation purpose in this study has been utilised.

Optimisation method: Based on the adoption of optimisation methods as reported in the literature, the heuristic and meta-heuristic approaches have gained major attention of the researchers compared to other optimisation methods. In this research, *hyper-heuristic search based* approach has been adopted since it has high potential to compute accurate and optimised results. Furthermore, hyper-heuristic approaches are mainly used to achieve better performance by integrating the features of existing meta-heuristic search-based algorithms, which would certainly minimise the completion time (makespan), and total computational cost challenges as a result of concurrent processing.

Structural representation: From in-depth analysis related to the adoption of structural representations, it has been found that majority of work have considered *Directed Acyclic Graph (DAG)* or the modified DAG model for structural representation. The proposed approach in this study has adopted DAG structural representation due to the potential to handle complex SWFAs in cloud computing systems. DAG provides means for defining precedence constraints relationships between workflow tasks for different sizes and types of SWFAs.

Scheduling technique: By analysing the scalability aspects reported in the literature, it can be clearly found that most of the work targeted dynamic approaches. In this study, the *dynamic scheduling* method has been adopted, since it requires no prior knowledge about the considered execution time and execution cost parameters of SWFS cost optimisation approaches.

Workload type: The *predicted* (batch mode) type of workload remains a key focus of researchers in this cost aspect of SWFS approaches. In this research, the predicted

workload type has been considered since it offers SWFS models with maximum throughput of the workload, and minimised turnaround time (the time between task submission and task completion).

Profitability: Based on the reviewed approaches, the profitability aspect of both *service consumer* and *service provider* should be consider in order to minimise the execution time and execution cost of SWFS in cloud computing. Therefore, by reducing the cost of maintaining resources, cost optimisation would be profitable for the service provider. Consequently, this provides an extra advantage for cloud service consumers since it would reduce the cost of workflow execution.

QoS constraints: Quality of Service (QoS) has a major impact on SWFS in cloud computing, since the success of execution of SWFA tasks heavily depends on the desired QoS levels. For the SWFS problem in cloud computing, there are several QoS constraints that must be taken into account for a given service when designing an efficient WfMS. This study has considered another important QoS aspect, the mechanism of handling the QoS constraints in SWFS approaches. The adopted method allows the users to assign *activity-level QoS* constraints, and then the overall QoS can be assessed by computing the QoS constraints of all individual activities based on the specific QoS model.

3.1.2 Cost Optimisation Parameters of SWFS

From the cost optimisation parameters' point of view, Figure 3.4 shows the options selected from analysing the two types of cost parameters: (i) monetary cost, and (ii) temporal cost. The details on about the monetary and temporal cost parameters have been explained in Section 2.2.

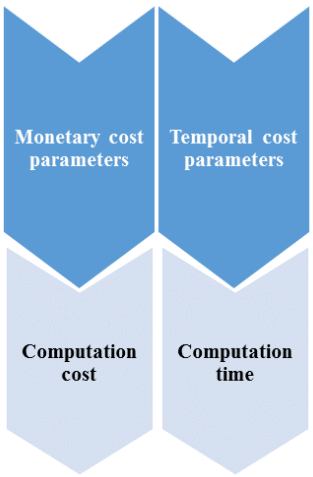

Figure 3.4: Cost optimisation parameters

By analysing the monetary cost parameters, it has been observed that the total monetary cost is based on the strong dependency between computation cost, and communication cost. This research has focused on the *computation cost*, since communication cost has a marginal impact due to the adoption of private cloud (where all the SWFA tasks are executed on the same computational environment). The storage cost is assumed to be of minimal value because of the usage of private (local) resources instead of renting the computational space from service providers (public). In contrast, by analysing the temporal cost parameters, it has been found that several approaches have considered measuring pre-scheduling parameters for their significance in determining estimated execution time, which requires the scheduler to handle the uncertainties challenges of SWFS algorithms. For instance, the scheduler requires to check that whether the submitted SWFS tasks are schedulable or not. The scheduler would make decisions based on a set of attributes: (i) number of available processors, (ii) list of available processors, (iii) currently used processors, (iv) name of processors, (v) name of computational site (VMs), and (vi) estimated execution time. Furthermore, historical data plays an important role in supporting the scheduler to optimally schedule the submitted SWFS tasks based on the availability of existing

computational resources. Therefore, there is a need to consider historical data in order to check the estimated execution time. Section 4.1.3.1 of Chapter 4 defines all the utilised parameters in the SWFS cost optimisation model.

3.1.3 Cost Optimisation Challenges of SWFS

From the cost optimisation challenges' point of view, cost optimisation challenges of SWFS have been affected by different workflow system structure and service-driven aspects. Figure 3.5 shows the options selected from the analysis of three main categories of challenges: QoS performance, system functionality, and system architecture aspects. Complete details on these aspects have been explained in Section 2.3.

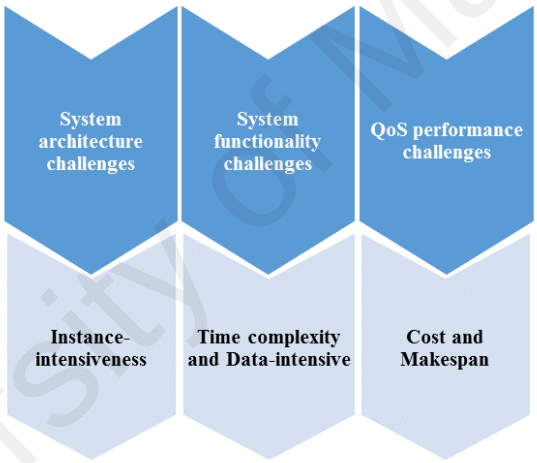

Figure 3.5: Cost optimisation challenges

By analysing the QoS performance challenges, it can be concluded that with respect to the QoS constraints of workflow execution processes, scheduling should adjust to the workflow model's constraints, such as cost constraints of the utility workflow process model or time constraints for the timed workflow process model, or both. Similarly, QoS performance challenges have direct effect on the completion time (makespan) and total computational cost, which are important for negotiating the process between the service consumers and service providers. It is evident that the majority of the work targeted

the makespan and total cost challenges. This is due to the fact that these challenges are related to SWFS execution time and execution cost, which are highly considered by service providers and also demanded by service consumers. This study mainly focuses on cost optimisation of SWFS by considering *completion time* and *total computational cost* as the QoS constraints.

By analysing the system functionality challenges, it can be concluded that a large number of approaches measuring time complexity requires finding a high quality solution in polynomial time from the available large search space. Therefore, this study considers measuring time complexity of the proposed SWFS cost optimisation approach. Additionally, this study also considers another important system functionality challenge, i.e. data-intensive and computational-intensive challenges. The *data-intensive and computational-intensive* challenges for scheduling SWFAs in cloud computing have received a major attention of researchers compared to other reported challenges in this category. This is mainly due to their correlation with the degree of data-intensiveness and scalability of the proposed approach.

By analysing the system architecture challenges, it is evident that the *intensiveness-based* challenges affect system distribution of WfMS while reducing execution time and execution cost. Similarly, intensiveness-related challenges play a vital role in the system architecture of WfMS in cloud computing environment. Several approaches focused on data-intensiveness for each system in WfMS. This requires large quantities of data computation and communication. However, the SWFS in cloud computing is currently facing a major challenge while handling only data-intensive applications, especially the data locality of application. Thus, this application ultimately needs more optimised solutions to handle this massive growth of SWFS data.

This study considers different sizes and types of SWFAs (e.g., Montage, Cybershake,

Inspiral, and Sipht) to evaluate the *data-intensive and computational intensive* challenges of the proposed approach.

3.1.4 Cost Optimisation Approaches of SWFS

The meta-heuristic search-based approaches from different perspectives (Chapter 2) has been discussed. The first perspective is about the types of meta-heuristic approach that have been proposed in the literature. The second perspective is about the strengths and limitations of each type of meta-heuristic approach and how the approach in this work has overcome these limitations. By analysing the meta-heuristic approaches, it has been observed that several types of meta-heuristic approach have been proposed for cost optimisation problem of SWFA in cloud computing environments as shown in Figure 3.6. The development of single-based solutions for cost optimisation problem of SWFS can introduce several strengths: (i) they can find a quick and acceptable scheduling solutions, (ii) they are easy and simple to be implemented in WfMS, and (iii) they can be integrated with other population-based approaches to obtain better solutions. However, the major limitation of a single-based solution is that it usually can not produce optimal solutions for problems that have a large search space, comparing with the result obtained by population-based solutions. This issue is also called local optima, which happens when the algorithm can not guarantee that a globally optimal solution can be found on the feasible search space of the scheduling problems.

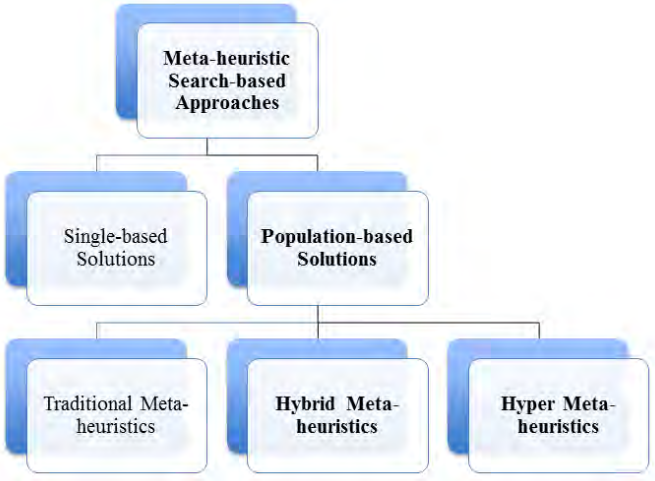

Figure 3.6: Meta-heuristic approaches

In addition, the traditional population-based meta-heuristic solutions have shown good performance for the optimisation problem having a large search space. This is because the traditional meta-heuristics do not exhaustively search the scheduling problem space. They used different underlying strategies to find the desired solution based on defined fitness criteria. Therefore, population-based (e.g., random-guided solutions) take less computational effort than single-based solutions and they can often find good solutions. However, each type of solutions has some strengths and limitations, which affects the scheduling operation of SWFS.

In contrast, the hybrid meta-heuristic used the best features of two or more traditional meta-heuristic meta-heuristic in each iteration to provide a better optimal solution compared to the traditional heuristics. However, in some cases and due to the complexity of hybrid meta-heuristic method, it might take a longer convergence time than the traditional meta-heuristic at each iteration process. Furthermore, the hybrid approaches could require a longer scheduling time. This study focuses on enhancing the existing *hyper-heuristic* approaches to optimise the makespan and total computational cost for cost optimisation problem of SWFS in cloud computing. The hyper-heuristic is an emerging class of

meta-heuristic search-based approaches that are combined in such a manner that allows utilising the maximum strengths of employed meta-heuristic algorithms to obtain an optimal solution. In most cases, hyper-heuristic approaches lead to the most optimised results. Furthermore, designing a hyper-heuristic algorithm by integrating the features of existing meta-heuristic search algorithms will certainly improve the cost optimisation challenge due to concurrent processing. However, there are few related works, which have considered enhancing of hyper-heuristic for cost optimisation of SWFS in cloud environments.

As an addition to the above mentioned details, there are also several important points that have been concluded from studying the strength of underlying techniques and the limitations of other approaches for cost optimisation of SWFS. The complete details about these points have been described in Section 2.4. These points have helped in proposing an approach for cost optimisation of SWFS in cloud computing environment.

3.2 Approach Development Stage

The guidelines and recommendations expressed in the formulation stage have helped in desinging and developing a cost optimisation SWFS approach. In this study, two main activities have been conducted to develop the proposed completion time driven hyper-heuristic approach, and the outcome defined in this study are: (i) a cost optimisation model of SWFS, (ii) the proposed dynamic hyper-heuristic algorithm for cost optimisation of SWFS in cloud environment. Complete details of the proposed approach will be presented in Chapter 4.

3.2.1 Cost Optimisation Model of SWFS

The cost optimisation model of SWFS mainly aims at mapping and managing the execution processes of the submitted dependent workflow tasks (also referred as precedence

constraints) of SWFA. The cost optimisation model schedules the submitted tasks on the targeted shared computational resources (VMs), while optimising the completion time and total computational cost. The standard cost optimisation model contains three main components: (i) scientific workflow application (type and size of SWFA), (ii) targeted computing environment (number of VMs), and (iii) cost optimisation criteria. There are three main categories of users that can utilise and manage the SWFS cost optimisation model: (i) service consumers (e.g., scientists in this research context) represent the users of the SWFA, (ii) service providers (i.e. cloud service provider) that offer the SWFS system with virtualised computational resources (i.e. private, public or hybrid), and (iii) system developers (i.e. system administrator) responsible for designing the WfMS based on the requirements of service consumers. The complete detail about each component of the cost optimisation model of SWFS has been provided in Chapter 4.

3.2.2 Proposed Dynamic Hyper-heuristic Algorithm

In this research, a dynamic hyper-heuristic algorithm for cost optimisation problem of SWFS in cloud computing environment has been proposed. The proposed algorithm employs four well-known population-based meta-heuristic algorithms, which act as Low Level Heuristic (LLH) algorithms (i.e., genetic algorithm, particle swarm optimisation, invasive weed optimisation, and hybrid invasive weed optimisation). In addition, the proposed algorithm enhances the native random selection way of existing hyper-heuristic algorithms by incorporating the best computed workflow completion time to, which act a high-level selector to pick a suitable algorithm from the pool of LLH algorithms after each run. The main aim of the proposed dynamic hyper-heuristic algorithm is to reduce the completion time and total computational cost to execute the SWFA. Based on the lowest achieved completion time, the proposed algorithm dynamically guides the searching processes to find an optimal solution. To achieve this, The proposed algorithm

continuously sorts the computed time scores (i.e. completion times of previous runs) of all LLH algorithms for each run of the considered scenarios. The computed time scores are listed in a scoreboard table. Next, for each single run, the high-level selector adopts the LLH algorithm that has the lowest computed time score for each scenario. The proposed dynamic hyper-heuristic algorithm continuously updates the scoreboard table by replacing the existing time score with the lowest computed time score, which ultimately affects the total computational cost value for that run. Finally, based on the scoreboard table, the proposed algorithm selects the most appropriate LLH algorithm for the next run. Consequently, the mechanism of the proposed dynamic hyper-heuristic algorithm becomes more effective in allowing to reuse and utilise the maximum strengths of the employed LLH algorithms. Ultimately, it helps in searching the optimal solution of the targeted cost optimisation problem. A complete detail of the proposed approach is provided in Section 4.2 of Chapter 4.

3.3 Evaluation and Analysis Stage

In this section, a methodology has been presented that is adopted to evaluate the proposed approach based on the guidelines of state-of-the-art cost optimisation approaches of SWFS.

3.3.1 Comparison of Existing Experimental Tools

In order to assess the quality of the proposed completion time driven hyper-heuristic approach for cost optimisation of SWFS in a cloud environment, several steps have been followed to study the most relevant approaches to identify the attributes and corresponding parameters from selected studies, also to extract the parameters' values from each defined attribute, as well as to construct a comparative table.

The following are the main perspectives that have been considered while analysing the existing experimental tools:

- Types of existing experimental cloud environments have been adopted in the reviewed studies.
- Number of computational resources have been utilised in the reviewed studies.
- Types of computational resources have been used to conduct the experiment.
- Types of SWFAs have been executed.
- The average size of the considered SWFA data sets has been considered.

After that, several attributes and the corresponding parameters from the reported studies have been identified. In total, six attributes were extracted: (i) name of the used tool environment, (ii) type of environment, (iii) number of computational resources, (iv) type of resource (based on Amazon instance specifications as a standard (*Amazon EC2 instance types*, 2015), (v) type of SWFA, and (vi) size of SWFA tasks. The researchers have considered a variety of computational resources, such as VMs, servers, and super computers, depending upon the selected tool environments and computational environment. Furthermore, a varying set of parameters for each extracted attribute have been identified. For instance, regarding the type of tool environment's attribute, two main parameters, including real-world experiment and simulation, have been used to develop these studies. Similarly, three scales for SWFA task attribute (i.e. small (<100 tasks), medium (100-1000 tasks), and large (>1000 tasks)) have been defined. Subsequently, a comparative table to map the attributes and corresponding parameters (see Table 3.1) has been developed.

Table 3.1: Qualitative comparison results of cost optimisation SWFS approaches

Reference	Name of tool environment	Type of tool environment	No. of resources	Type of resource	Type of SWFA	No. of tasks
(Z. Wu et al., 2013)	SwinDeW-C based on Hadoop	Real-world experiment (empirical study)	30	t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium
(Prodan & Wieczorek, 2010)	Invmod, Austrian Grid, and Water Balance Simulation Model ETH (WaSiM-ETH)	Simulator (open source) and Real-world experiment (Case study)	16	M3.large, M3.extra-large	parallel, random, and CSTEM	Medium
(Abrishami & Naghibzadeh, 2012)	Developed tool in Java	Simulator	9	t2.small, t2.medium, t2.large, S3	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large
(Ramakrishnan et al., 2011)	TeraGrid	Batch experimentation (empirical study)	10	t2.small, t2.medium, t2.large	Lead (weather), Motif (storm-surge)	Large
(Saeid Abrishami, 2012)	GridSim and DAS-3	Simulator (open source)	272	t2.small, t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT.	Small-Medium-Large
(Nargunam & Shajin, 2012)	CloudSim	Simulator (open source)	9	t2. micro, t2.small, t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small
(W.-N. Chen & Zhang, 2009)	Project scheduling problem library	Real-world experiment (Empirical study)	10	t2.small, t2.medium, M4.large	10 types of scientific WFs	Medium-Large
(Q. Wu et al., 2013)	Fair-share scheduling policy in C++	Simulator, and real-world experiment (Case study)	6	t2.medium	Weather research and forecasting (WRF)	Small-Medium-Large
(Ostermann et al., 2010)	ASKALON, GroundSim, and Eucalyptus middleware in Java	Simulator (open source)	5	M1.large, C1.xlarge	Wien2k, Invmod, and meteoRG	Medium
(Y. Yuan et al., 2006)	Developed tool	Real-world experiment	3	N/A	serial and parallel	Small
(H. Liu et al., 2011)	Developed tool in Java	Simulator	10	t2.medium	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large

Continued on next page

Table 3.1 – continued from previous page

Reference	Name of tool environment	Type of tool environment	No. of resources	Type of resource	Type of SWFA	No. of tasks
(J. Yu & Buyya, 2006b)	GridSim and GridSim Index Service (GIS)	Simulator (open source)	15	N/A	neuro-science, EMAN, protein annotation Montage	Small
(Sakellariou & Zhao, 2004b)	Rescheduling policy	Simulator (open source)	8	N/A	Fork-Join, Laplace equation solver, FFT	Small-Medium
(J. Yu & Buyya, 2006a)	GridSim and GridSim Index Service (GIS)	Simulator (open source)	4	t2.small, t2.medium and service type (*Align_wap* and *Reslice)*	neuro-science workflow, Hybrid structure, protein annotation	Small
(Afzal et al., 2006)	ICENI, GridSim toolkit, Mixed-Integer Non-linear programming (MINLP)	Simulator	24	M3.large, M3.extra-large	N/A	Large
(Y. Yang et al., 2008)	SwinDeW-C based on Hadoop	Simulator	8	N/A	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large
(Stevens et al., 2009)	Phosphorus network and CloudSigma	Simulator	3	C4.8 extra-large	N/A	Large
(Sakellariou et al., 2007)	Developed tool	Real-world experiment	3	N/A	FFT, Fork-Join, Laplace, Random DAGs	Medium
(Genez et al., 2012)	Java and IBM ILOG CPLEX Optimiser	Real-world experiment	4	M3.2 extra-large	Montage fork-join DAG	Small
(Ranaldo & Zimeo, 2009)	IEEE 118-node, UDDI, WSDL, GonQoS, and CFM1	Real-world experiment (Case study)	8	t2.micro	N/A	Small-Medium
(Saeid Abrishami, 2013)	Developed tool and Amazon EC2	Simulator (open source)	10	t2.small, t2.medium, t2.large	Montage, cyberShake, Inspiral, Sipht, Epigenomics	Small-Medium-Large
(J. Yu et al., 2005)	GridSim, and GridSim Index Service (GIS)	Simulator (open source)	15	N/A	pipeline, parallel and hybrid-*Align_wap* and *reslice*	Small

Continued on next page

Table 3.1 – continued from previous page

Reference	Name of tool environment	Type of tool environment	No. of resources	Type of resource	Type of SWFA	No. of tasks
(Talukder et al., 2009)	Developed tool	Real-world experiment	10	N/A	Gauss–Jordan Algorithm, LU decomposition, Laplace Transform	Small-Medium
(Pandey et al., 2010)	JSwarm package, and Amazon CloudFront	Real-world experiment (Empirical study)	3	t2.small, t2.medium, t2.large	N/A	Large
(Tanaka & Tatebe, 2012)	Pwrake workflow system, InTrigger Kore, Gfarm distributed file system in Ruby language	Real-world experiment (Empirical study)	8	M4.extra-large, M4.10 extra-large	Montage 2MASS	Large
(Bittencourt & Madeira, 2011)	Amazon Elastic Compute Cloud	Real-world experiment (Empirical study)	3	t2.small, t2.large, M4. extra-large	Montage, AIRSN, CSTEM, LIGO-1 and LIGO-2, Chimera-1, Chimera-2, median filter image processing	Small-Medium
(Srinivasan, 2012)	SwinDeW-C, and Hadoop	Real-world experiment (Empirical study)	8	N/A	Montage, cyberShake, Inspiral, Sipht, Epigenomics	Small-Medium-Large
(Shi, 2014)	AMS	Real-world experiment (Empirical study)	5	t2.small, t2.medium, t2.large	Stable, On-and-Off, Crowing, Bursting	Small-Medium-Large
(Viana et al., 2011)	SciCumulus	Developed tool in Java	8	M4.extra-large	DNA sequences	Small-Medium
(Malawski et al., 2012)	CloudSim	Simulator (open source)	9	t2.small, t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large
(Zhu et al., 2012)	Developed tool in C++	Real-world experiment (Test cases)	15	t2.medium	N/A	small-Large
(X. Lin & Wu, 2013)	CloudSim	Simulator (open source)	9	t2.small	VM-Amazon EC2	Small-Medium-Large
(Tsai et al., 2014)	CloudSim and Hadoop	Simulator and Real-world	5	t2.small	grep, log analyzer, TeraSort	small-Large

Table 3.1 presents the comparison results of reviewed approaches. It can be clearly seen that researchers have mainly considered simulation based environment compared to the real-world based environment. Availability of tools justifies this trend, since standard dataset in terms of open source (simulation) is easily available compared to the real-world experimentation. Regarding the number of resources' attributes, several types and specifications were utilised due to the need for heterogeneous computational resources. It is evident that the majority of the cost optimisation SWFS approaches used a large size of computational resources due to the nature of some SWFAs. As for the type of SWFA, several types have been used such as Montage, CyberShake, Inspiral, and SIPHT (*Pegasus workflow repository-Workflow Generator*, 2015). Moreover, most of the considered SWFAs have used different types of tasks dependencies (e.g., process, pipeline, data distribution, data aggregation, or data redistribution). Not all the approaches have considered the three scales of tasks (i.e. small, medium, large). Therefore, to effectively measure the data-intensiveness and computational-intensiveness performance of cost optimisation SWFS approaches, it is important to consider the different sizes of SWFA tasks.

3.3.2 Experimentation Setup of the Proposed Approach

By analysing the existing experimental tools environment (Section 3.3.1), it has been found that there is a need to consider different experimental environments. As a result, it helps in developing and evaluating the performance of any cost optimisation approach of SWFS in cloud computing environment. In this thesis, two different experimental cloud environments have been considered: (i) a simulation-based environment using WorkflowSim, (ii) and a real-world based environment using Pegasus workflow management system.

3.3.2.1 Simulation Based Environment

The first type of evaluation for the proposed approach is conducted through a WorkflowSim simulator by setting it up on a eclipse editor. The WorkflowSim simulator has been used to develop and evaluate the proposed completion time driven hyper-heuristic approach for cost optimisation of SWFS in a cloud environment. WorkflowSim simulator used the data collected from executing the actual SWFA to generate synthetic workflows resembling those which are used by real-world scientific applications. Simulation-based environment helped in clearly understanding the scheduling process. At the same time, it facilitates by easily determining different scenarios (e.g., number of VMs, size of SWFA) to completely investigate the performance of the proposed approach. WorkflowSim simulator is an open-access programming tool for developing and simulating the cloud based workflow systems by supporting a parallel and distributed infrastructure. WorkflowSim simulator has the required CloudSim techniques that support the execution of workloads in cloud computing infrastructures and services. CloudSim classes and packages have been used to read the task values for the workload traces. A complete detail on the evaluation of the proposed approach in the simulation environment is provided in Chapter 5.

3.3.2.2 Real-world Based Environment

Workflow Management System (WfMS) in cloud computing must have the ability to handle the requests from different application domains of SWFAs. In this study, the Pegasus WfMS has been utilized as the second type of environment for evaluating the proposed approach. The HTCondor and Pegasus WfMS are used to represent and manage the complex dependent tasks and related data files of SWFAs. Pegasus has a number of features that contribute to its usability and effectiveness including portability, reusability, reliability, scalability, performance, data management, and error recovery.

A complete detail on the evaluation of the proposed approach in this real-world test-bed

environment is provided in Chapter 6.

3.3.3 Baseline Approaches

As it has been explained earlier in this chapter (Section 3.1.4). This study focused on comparing the proposed completion time driven hyper-heuristic approach of SWFS with the most relevant population-based algorithms (i.e. GA, PSO, IWO, HIWO) and an existing hyper-heuristic approach named Hyper-Heuristic Scheduling Algorithm (HHSA)to effectively and efficiently evaluate the performance.

In the simulation environment (WorkflowSim), to comprehensively evaluate the proposed approach with the existing cost optimisation problem of SWFS algorithms, the populationbased algorithms have been considered GA, PSO, IWO, HIWO and HHSA, as comparison baseline approaches. This is mainly due to the reason that these approaches achieved significant results in the previously conducted studies (i.e. GA (J. Yu & Buyya, 2006b, 2006a; X. Liu, 2011), PSO (Q. Tao et al., 2009; Jing et al., 2013; Pandey et al., 2010; B. Liu et al., 2014; Z. Wu et al., 2010; Netjinda, Sirinaovakul, & Achalakul, 2012), IWO, (K. Li, Li, Xu, & Xie, n.d.; Sharma, Nayak, Krishnanand, & Rout, n.d.; X. Zhang, Niu, Cui, & Wang, n.d.; Sang & Pan, n.d.). In Appendix A, more details about each of these baseline and HHSA approaches has been provided.

In the real-world environment, the proposed approach has also been compared with other baseline approaches (i.e. GA, PSO, IWO, HIWO and HHSA) in Pegasus SWfS to comprehensively evaluate the proposed approach with the existing algorithms. Note that current algorithms have been already developed for cost optimisation problem of SWFS in cloud environment.

3.3.4 Scientific Workflow Application Datasets

The proposed approach has been evaluated based on different types of SWFA datasets namely Montage, Cybershake, Inspiral, and Sipht. These applications have been selected since they have been extensively used in the prior work (Juve et al., 2013; W. Chen, da Silva, Deelman, & Sakellariou, 2015; W. Chen, Ferreira da Silva, Deelman, & Fahringer, 2015; Jrad, Tao, Brandic, & Streit, 2014; W. Chen, Da Silva, Deelman, & Sakellariou, 2013). Moreover, Chen et. al. (W. Chen, da Silva, et al., 2015) has regarded that these SWFAs act as standard datasets to evaluate the performance of the existing approaches. Furthermore, these applications have different sizes that allow measuring the scalability of data-intensive applications. Moreover, they consider the computational-intensive of the SWFA tasks, which can be measured by the number of compositions in their structure. The following URL link contains the real-world Montage datasets:

https://drive.google.com/file/d/0B1X3lOeJffFIM00xeHVqREtCbDg/view?usp=sharing

For more details about each of the used scientific workflow application datasets, please refer to Section 5.3, Section 6.3, and Appendix B.

3.3.5 Statistical Analysis

The proposed completion time driven hyper-heuristic approach for cost optimisation of SWFS in a cloud environment has been implemented using two types of cloud environments which are WorkflowSim (simulator) and Pegasus WfMS (real-world test-bed). In the simulation-based environment, the collected data from executing each of the four scientific workflow datasets (i.e. Epigenomics, Inspiral, Montage, Sipht) using three different sizes of SWFA datasets. In contrast, in the real world-based computational environment, real SWFAs have been used to evaluate the proposed approach. All the collected data have been saved in Microsoft Excel sheets. After that, the collected data has been imported into IBM SPSS statistics tools version 24. Then, four statistical methods were performed in

this research in order to perform a complete analysis of the collected data. The descriptive analysis test is particularly useful to get a general observation of the data collected. Four main values have been chosen for the completion time (makespan) and total computational cost, which are standard deviation, mean value, maximum value (represents the worst value), and minimum value (represents the best value).

To measure the significance improvement of the result, there is a need to statistically investigate the normality tests to determine whether it is normal or not (Wohlin et al., 2012). There are couple of different ways to determine the distribution of the data for the result data of completion time and total computational cost. Furthermore, in order to decide whether the data is paired or unpaired, the normality test of the results has been conducted. The Skewness and Kurtosis values help to investigate the normality distribution of the data. If the Skewness and Kurtosis values are between range of -2 and 2, then the result data is considered as normally distributed. The other two important normality tests are Kolmogorov-Smirnov and Shapiro-Wilk tests, which have been widely considered by researchers to confirm the normality distribution. The p-value of Shapiro-Wilk has to be greater than 0.05, which specifies that the results data are normally distributed.

3.4 Summary

In this chapter, a broad overview of the defined research methodology stages (i.e. formulation stage, approach development stage, and evaluation and analysis stage) has been provided. In the formulation stage, an in-depth analysis by conducting extensive literature review from different cost optimisation perspectives of SWFS (i.e. aspects, parameters, challenges, and approaches) has been provided. The results from in-depth studying and analysing the targeted perspectives have guided towards the approach development stage as well as the evaluation and analysis stage. In the approach development stage, a cost optimisation model as well as the dynamic hyper-heuristic algorithm have been

comprehensively defined which would be used in the proposed approach. In the evaluation and analysis stage, five major activities have been conducted to evaluate the proposed completion time driven hyper-heuristic approach for cost optimisation of SWFS in a cloud environment. The activities are: (i) the comparison of existing experimental environments, (ii) the experimentation setup of the proposed approach, (iii) the selection of baseline approaches, (iv) the consideration of the scientific workflow application datasets, and (v) the execution of statistical analysis for the collected data.

CHAPTER 4: PROPOSED APPROACH

In this chapter, the proposed approach is presented to successfully accomplish the second stage of the defined research methodology. The proposed approach aims at the most challenging problem with Scientific Workflow Scheduling (SWFS), which is the cost optimisation of SWFS in a cloud computing environment. The cost optimisation challenge of SWFS in cloud computing requires considering several scenarios that depend on the number of available virtual machines and size of SWFA datasets. Current state-of-the-art SWFS approaches lack in achieving satisfactory cost optimisation performance for all considered scenarios. Therefore, there is a need to propose a completion time driven hyper-heuristic approach that can effectively optimise the cost of SWFS for all considered scenarios. The proposed approach contains two main parts, the cost optimisation model of SWFS and the dynamic hyper-heuristic algorithm (Figure 4.1).

The cost optimisation model of SWFS can help to understand the mapping and scheduling processes of workflow tasks by considering the scheduling stages along with completion time and total computational cost parameters. Three standard stages of SWFS cost optimisation model are defined in this chapter. The first stage concerns about the SWFAs, while the second stage concerns about the targeted computing environment. Finally, the third stage concerns about formulates the cost optimisation criteria.

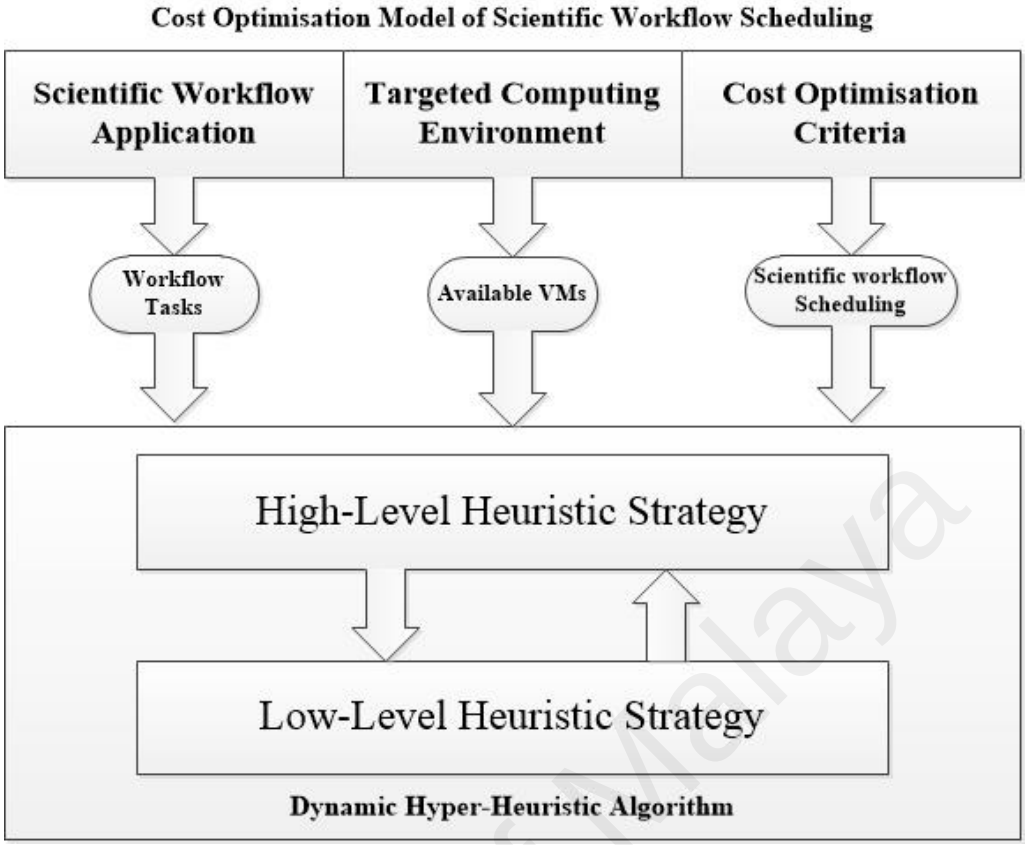

Figure 4.1: Completion Time Driven Hyper-Heuristic approach

After defining the SWFS cost optimisation model, the proposed dynamic hyper-heuristic algorithm for the cost optimisation challenge of SWFS in cloud environment has been presented. The proposed algorithm is considered as a new advanced technique that is capable of reducing the run-time of a meta-heuristic algorithm. The proposed dynamic hyper-heuristic algorithm contains four parts, where these parts represent the four main stages of the proposed algorithm.

4.1 Cost Optimisation Model of Scientific Workflow Scheduling

The cost optimisation model of SWFS mainly aims at mapping and managing the execution processes of the submitted dependent workflow tasks (also referred as precedence constraints) of SWFA. It then schedules the submitted SWFA tasks on the targeted shared computational resources, while optimising the completion time and the total computational

cost. The standard cost optimisation model contains three main stages: (i) scientific workflow application, (ii) targeted computational environment, and (iii) cost optimisation criteria.

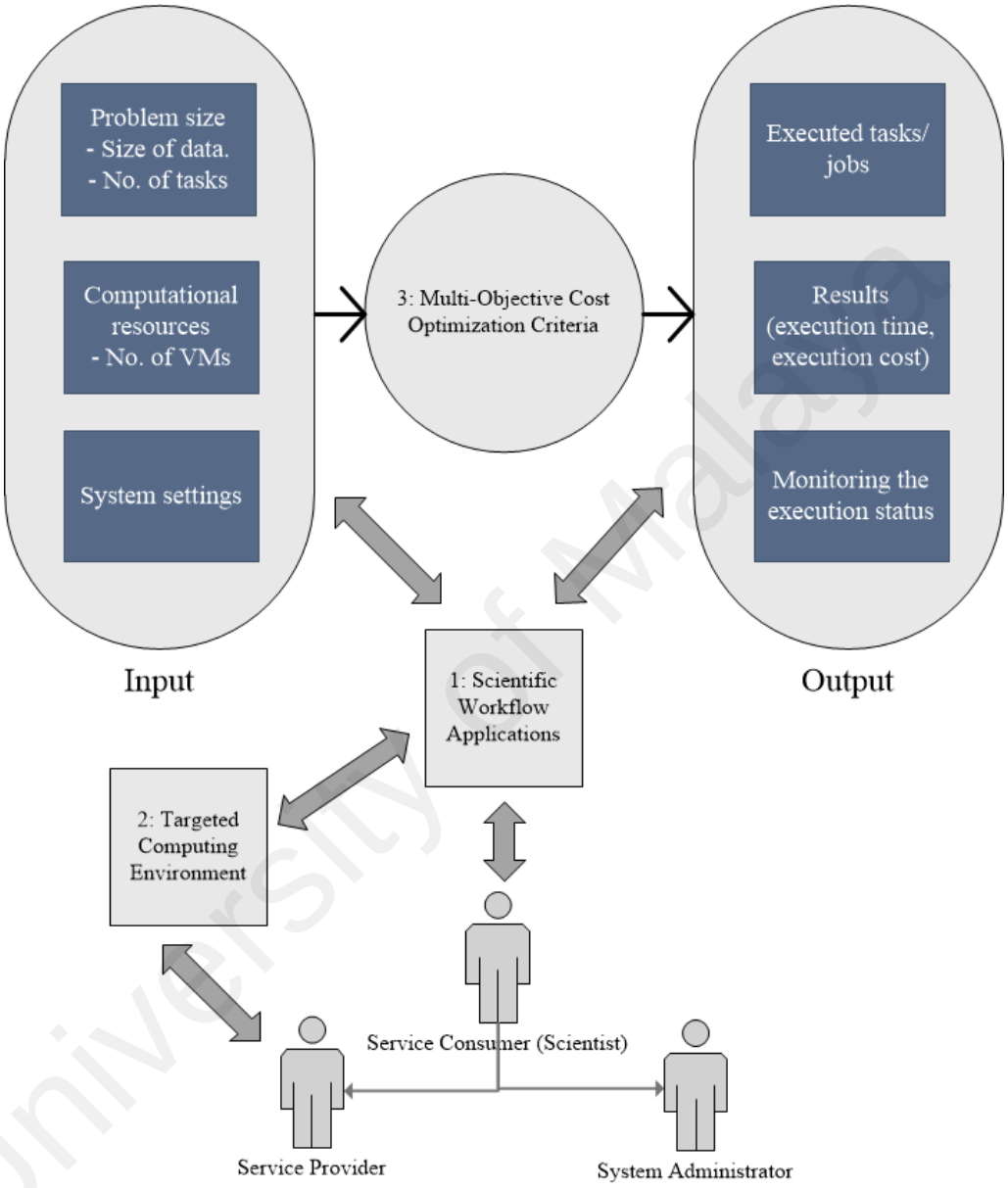

Figure 4.2: Cost optimisation model of SWFS

As shown in Figure 4.2, there are three main categories of users that can utilise and manage the cost optimisation model: (i) service consumers (i.e. scientists in this research context) represent the users of the SWFA, (ii) service providers (i.e. cloud service providers) that offer the SWFS system with virtualised computational resources (i.e. private, public or

hybrid), and (iii) system developers (i.e. system administrators) responsible for designing and maintaining the WfMS based on the requirements of service consumers. In the following sections, a complete detail about each component of the cost optimisation model of SWFS will be provided.

4.1.1 Scientific Workflow Application

The SWFA is also known as data and computational intensive scientific workflow application, which mostly processes data flows together with the tasks execution (Wieczorek et al., 2009; Ma et al., 2009; Malawski et al., 2014; Tolosana-Calasanz et al., 2012). The SWFA consists of multiple tasks, which are necessary to complete a particular submitted workflow. The components of these tasks can be any executable instances (e.g., load sets, report sets, programs, and data) with different structural dependencies (e.g., process, pipeline, data distribution, data aggregation, and data redistribution). The SWFA includes various input scripts (e.g., scientific program along with their dependent data), which can be used to produce, analyse and visualise the obtained results (Figure 4.2). Moreover, the SWFA has to deal with a large size of data and complex workflow tasks (e.g., earthquake prediction applications, biomedical applications, and astrophysics applications). The output of SWFA provides interactive tools that help service consumers in better executing their own workflows, and also visualising the results in a real time manner. At the same time, the SWFA simplifies the execution processes for scientists to reuse the same workflows. Furthermore, the SWFA provides an easy-to-use environment to track and share output results virtually. However, high dependency between the workflow tasks remains a prominent challenge of SWFA, which arises due to the tasks precedence constraints. Thus, SWFAs require extra powerful computational resources to efficiently determine an optimal SWFS solution for large data and complex tasks. Due to the complex nature of SWFA, the structural representation plays an important role to simplify the submitted

workflow tasks of SWFA. Thus, it is important to perform the modeling for the submitted tasks along with their precedence constraints using standard notations. In the literature, several types of structural representation methods have been adopted to represent the tasks' dependency of SWFA. One of the commonly used structural representation method is a Direct Acyclic Graph (DAG) that can handle SWFA tasks with high complexity. The DAG is beneficial to highlight the estimated execution cost of different available resources based on the historical data of WfMS. In addition, the communication time between resources is also represented using DAG (Z. Wu et al., 2013; Pandey et al., 2010; Talukder et al., 2009; Lombardi & Di Pietro, 2011; X. Liu et al., 2011). The DAG is represented by a graph (G) as formulated in Eq. 4.1.

$$G = (T, E) \tag{4.1}$$

where T (vertex) is a set of tasks and E (edges) is a set of directed edges between the tasks.

$$T = \{t_0, \ldots, t_n\} \tag{4.2}$$

$$E = \{e_1, \ldots, e_m\} \tag{4.3}$$

Note that there is data dependency between tasks in E. For instance, if there is a directed edge e_{t_i,t_j} (i.e. $e_{t_i,t_j} \in E$) connecting t_i and t_j (denoted as $t_i \rightarrow i_j$) then t_i is considered as a parent and t_j as a child. Note that a child task cannot be executed until all of its parent task(s) have been executed successfully. Therefore, the input data of task j depends on the

produced data by parent task i. The complete path from t_0 to t_n can be represented in Eq. 4.4.

$$(t_0 \rightarrow t_1, (t_1 \rightarrow t_2, \ldots, (t_{n-2} \rightarrow t_{n-1}, (t_{n-1} \rightarrow t_n) \tag{4.4}$$

4.1.2 Targeted Computing Environment

In order to execute the workflow tasks in a cloud computing environment, it requires the mapping between the submitted tasks to the available set of heterogeneous/homogeneous computational resources. In the context of cloud computing, the computational resources are commonly referred as a set of Virtual Machines (VMs) as defined in Eq. 4.5.

$$VM = \{VM_0, ..., VM_k\} \tag{4.5}$$

where VM_k is a single virtual machine (i.e. $VM_k \in VM$) having different specifications in terms of processor speed, memory capacity, and hard disk space. Moreover, each virtual machine could be created and distributed on different virtual host locations. Note that each VM specification has different computational cost and communication cost between the available VMs.

In order to clarify the SWFS process, Figure 4.3 illustrates a DAG structure with 13 tasks (i.e. from vertex 0 to vertex 12), which are represented as nodes running on a set of VMs. The entry task usually holds an input file (*f.in*) and the exit task produces the output file (*f.out*. The number associated with each edge (i.e. represented using an edge weight) is the time required to transfer the data from parent node to child node. For instance, the

edge weight between vertex ‘o’ and vertex ‘1’ represents that 17 seconds (based on the SWFA type) are required to transfer the dependent data to any of the available VMs.

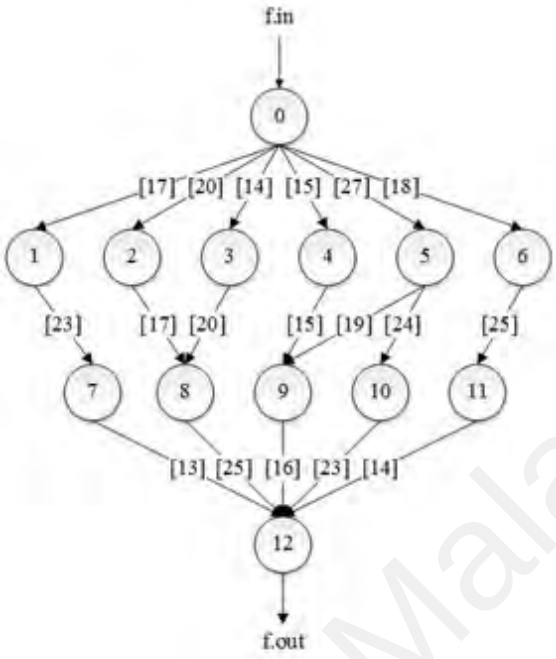

Figure 4.3: Topology of DAG

By doing this, each parent task generates output data (e.g., processed images) after the execution is completed and the child task(s) consumed the generated output data. Table 4.1, shows an example of the estimated execution time of each task based on the historical data.

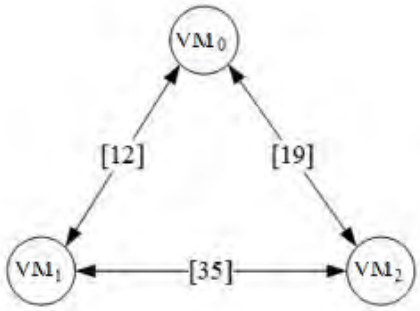

Figure 4.4: Representation of VMs dependencies

Figure 4.4 shows an example of the dependencies and the estimated communication time

Table 4.1: Example of the estimated execution time on VMs

Task	VM_0	VM_1	VM_2
0	8	40	23
1	8	43	15
2	12	22	27
3	35	16	12
4	3	12	16
5	26	4	23
6	23	21	16
7	22	9	17
8	5	11	15
9	14	20	34
10	8	18	11
11	19	29	7
12	13	33	3

Table 4.2: Example of estimated communication time between VMs

Connected VMs	Estimated Communication Time
VM_0 and VM_1	12
VM_0 and VM_2	19
VM_1 and VM_2	35

(in Sec.) between each pair of the VMs that are interconnected using various bandwidths as illustrated in a tabular format using Table 4.2.

4.1.3 Cost Optimisation Criteria

From the literature, it has been found that some work has focused on minimising the execution time (Nargunam & Shajin, 2012; Ramakrishnan et al., 2011; Stevens et al., 2009; Tanaka & Tatebe, 2012), while other work aimed at reducing the total computational cost (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013; H. Liu et al., 2011; Genez et al., 2012) for processing SWFA in cloud computing environment. The approaches that consider either execution time or execution cost (but not both) are commonly referred as single-objective optimisation based approaches. Due to the rapid development of the provided cloud computing services (e.g., pay-as-you-go, on-demand), many other

constraints (e.g., QoS constraints) should also be considered while optimising the cost of SWFS. The total computational cost of the submitted SWFA tasks is directly proportional to the time spent on computing the given tasks using the available computational resources. Due to the consideration of rapid development, the complexity of the existing approaches has increased to a great extent, which ultimately demands to carefully handle the trade-offs between the cost and other affected constraints (e.g., execution time). Nevertheless, a group of cloud services may have different specifications so they can complete the submitted workflow tasks with different execution time and execution cost based on the user requirements. For instance, the total computational cost is the amount of money to be paid for using the cloud computational resources (i.e. VMs). The computational cost can be categorised based on VMs specification (e.g., speed of CPU, size of memory and size hard disk). Moreover, each category (i.e. small, medium, large) of VM has a different price.

In order to propose a cost optimal solution for the SWFS problem, the completion time (makespan) and total computational cost of workflow tasks need to be minimised (Talukder et al., 2009). The cost optimisation of SWFS can be achieved by simultaneously minimising the execution time and execution cost. Thus, there is a need to consider applying Pareto-optimal solution method (Stephanakis et al., 2013; Yassa et al., 2013; Talukder, Kirley, & Buyya, 2007; Duan, Prodan, & Li, 2012; Durillo, Nae, & Prodan, n.d.). The underlying concept of Pareto-optimal method is to consider many solutions in the feasible region rather than focusing on a single solution. Based on the nature of optimisation problem, there are two types of objective, which are minimising or maximising the optimisation problem. In this research, minimizing completion time (makespan) of the submitted workflow tasks and total execution cost of workflow tasks has been considered as an optimization problem.

In this study, the standard SWFS cost optimisation model has been followed, which is based on the defined time and cost parameters as presented in Chapter 2 (Section 2.2). As shown in Figure 4.5, the cost parameters has been classified into three categories (i.e. pre-scheduling, during scheduling and post-scheduling).

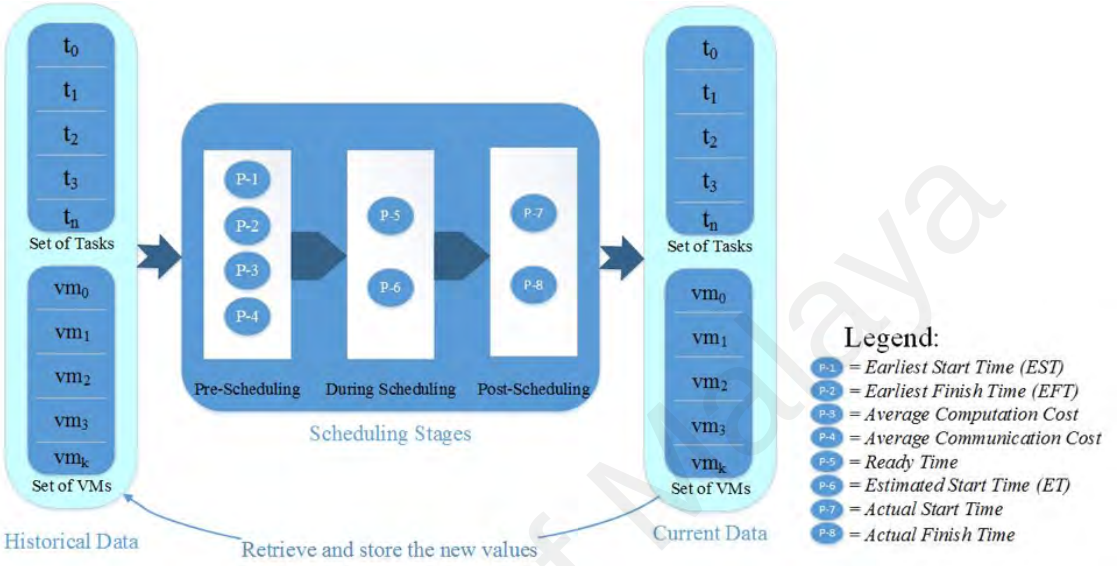

Figure 4.5: The considered cost-optimisation parameters

Pre-Scheduling Stage: In this scheduling stage, the scheduler requires to check whether the submitted SWFS tasks are schedulable or not. The scheduler makes a decision based on a set of attributes: (i) number of available processors, (ii) list of available processors, (iii) currently used processors, (iv) names of processors, (v) name of computational site, and (vi) estimated execution time. The historical data plays an important role in supporting the scheduler to optimally schedule the submitted SWFS tasks based on the availability of existing computational resources. Therefore, there is a need to consider the historical data in order to check the estimated execution time.

A computational cost matrix (w) of size t × VM is used to assign a particular task (t_i) to the available virtual machine (VM_k) by determining each estimated cost weight ($w_{i,j}$). Additionally, at pre-scheduling stage, the submitted workflow tasks are labeled with the

average computational cost. The average computation cost of a task t_i is defined by Eq. 4.6 (Poola et al., 2014):

$$w_i = \sum_{j=1}^{VM} w_{i,j}/VM_j \tag{4.6}$$

Similarly, the communication cost matrix ($\propto$) of size VM $\times$ VM represents the stored data's transfer rates between virtual machines ($data_{i,j}$). The communication startup costs of virtual machines are given in a VM-dimensional vector (L. The communication cost (c) of an edge *(i,j)* representing the data transfer rate from task t_i (scheduled on VM_m) to task t_j (scheduled on VM_n), is defined as (Arabnejad & Barbosa, 2014):

$$c_{i,j} = L_m + \frac{data_{i,j}}{\propto_{m,n}} \tag{4.7}$$

Consider a scenario in which both tasks t_i and t_j are scheduled on the same virtual machine. The value of communication cost ($c_{i,j}$) becomes zero, since the inter-processor communication cost inside the VM is negligible, which can be ignored. The average communication cost of an edge *(i,j)* is defined as follows (Arabnejad & Barbosa, 2014):

$$c'_{i,j} = L'_m + \frac{data_{i,j}}{\propto'_{m,n}} \tag{4.8}$$

where $\propto'$ is the average transfer rate among the virtual machines reside at the host site, and the L' is the average communication startup time. It is worth to mention that the average value has been considered by many researches in this area of research to support

the heuristic decision (S. Abrishami&Naghibzadeh (2012); M. E. D. H. Abrishami Saeid Naghibzadeh (2013)).

As already discussed in Section 2.2, based on the historical data and to efficiently start the scheduling processes, it is necessary to consider the Earliest Start Time (EST) and Earliest Finish Time (EFT) of the execution processes. The EST is defined as the earliest time to initiate the task execution on available VM (Stevens et al. (2009); S. Abrishami et al. (2012)). Nevertheless, it is impossible to exactly measure the EST in a heterogeneous environment, as a specific cloud's computation time of tasks differs within each virtual machine (S. Abrishami et al. (2012)). Every task has an estimated time period and should not be scheduled earlier than the EST, and must end latest by the Finish Time (Ramakrishnan et al. (2011)). The EST of each unscheduled task is already described in Eq. 2.1 and Eq. 2.2 in Chapter 2. In contrast, the EFT of each unscheduled task is the earliest time the task's execution can finish (S. Abrishami&Naghibzadeh (2012); M. E. D. H. Abrishami Saeid Naghibzadeh (2013)). Thus, it is essential to first calculate the EST, and then calculate the EFT for each task in the workflow prior to assign the SWFS tasks to the fastest and available computational resource (M. E. D. H. Abrishami Saeid Naghibzadeh (2013)). The EFT can be calculated using the Eq. 2.3 in Chapter 2.

During Scheduling Stage: It is of vital importance to check the ready time parameter in this scheduling stage. The ready time is the time by which all the data needed by the tasks has reached (after the parent(s) node has/have been executed) to the scheduled virtual machine (computational site). Therefore, ready time is defined as the earliest time for the first task to be executed and the first task is chosen based on the parent tasks (J. Yu & Buyya, 2005). Eq. 2.8 in Chapter 2 is used to calculate the ready time of task t_i. Additionally, the Estimated start Time (ET) is also considered as a minimum available time of the computational site.

Post-Scheduling Stage: After a task t_i is scheduled on a virtual machine VM_k, the EST of this execution is equal to the actual start time while the EFT of the execution is equal to the actual finish time of t_i. The time between the startTime and endTime represents the allocated time frame on the virtual machine for task execution (Stevens et al. (2009); J. Yu & Buyya (2006b); Sakellariou & Zhao (2004b)). The completion time of the entire workflow is based on parallel and serial constraints between the startTime and endTime (Y. Yuan et al. (2006)).

Once the submitted tasks are scheduled, the start time of the parent task will be used as a deadline for other dependent tasks (S. Abrishami & Naghibzadeh (2012)). There are two scenarios of tasks' start times in the scheduling algorithms. The first scenario is that the start time is EST, which is calculated prior to the workflow being scheduled. In contrast, the second scenario considers that the real start time is calculated after scheduling the tasks (M. E. D. H. Abrishami Saeid Naghibzadeh (2013); S. Abrishami et al. (2012)). The actual finish time represents the time that has been used to complete the execution of submitted workflow task (Sakellariou et al. (2007); Q. Wu et al. (2013); Sakellariou & Zhao (2004b)). In other words, after all the tasks are scheduled, the scheduling length (completion time) is represented as the actual finish time of the exit t_{exit}. If there are multiple exit tasks, the completion time of the scheduling (makespan) is considered as the maximum actual time of the exit task.

4.1.3.1 Cost Optimisation Parameters

The main goal of this research is to optimise the execution time and execution cost by minimising the completion time (makespan) and total computational cost of SWFA in cloud computing. The required execution time and execution cost of the available computational resources are related to each other. Consider a scenario which contains a number of VMs having different specifications (heterogeneous), and each of the VMs requires different

execution time and execution cost to execute the same set of workflow tasks. Therefore, the scheduler should choose the VM that takes a longer time but with cheaper cost, or the scheduler should choose the VM that takes a shorter time but incurs more expensive cost. Figure 4.6 shows an example of the inversely proportional relationship between the execution time and execution cost of VMs with different specifications.

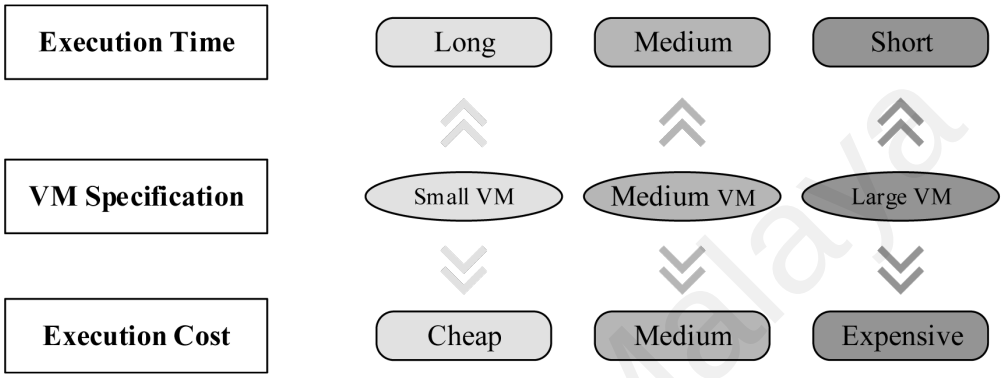

Figure 4.6: Correlation between the execution time and execution cost

Based on the discussed cost and time parameters, the main goal of the SWFS cost optimisation model in this research context is to minimise the completion time (makespan) and the total execution cost of the submitted SWFA. Thus, in order to propose an optimal solution for the SWFS problem, the total cost and time of executing workflow tasks needs to be minimised. However, there is no single optimal solution but rather a set of potential solutions for each considered scenario. Therefore, this research focuses on optimising two critical cost parameters of SWFS in a cloud computing environment, computational cost and completion time.

Completion Time (makespan): Makespan (M) can be defined as the overall completion time to execute the entire workflow by considering the finish time of the last completed task. Generally, any delay in task execution time can negatively impact on the makespan of workflow application. In the case of traditional workflow scheduling, the users generally choose to minimise the workflow makespan. Thus, the reported cloud-based SWFS

approaches focus on how to assign tasks to resources so that the precedence constraints are retained while the makespan is minimised. Hence, the major aim of makespan is to reduce the execution cost within a lower completion time while meeting the user's requirements effectively. This parameter is important due to its relevancy to QoS. The makespan of the workflow is computed as follows (Poola et al., 2014):

$$M = t_{exit} - ST \tag{4.9}$$

where ST is the Submission Time of the workflow and t_{exit} is the finish time of the exit node.

Total Computational Cost (TCC): The computational cost of a SWFA remains a main concern of the user(s). The computational cost of workflow tasks is directly proportional to the time spent on computing workflow tasks using the available cloud resources (Jing et al., 2013). For instance, the computational cost of the public cloud is represented by the amount of money to be paid for using the enterprise's computation resources, and the computational resources (VM) are categorised by a range of computing resource costs. Therefore, the total execution cost is calculated by summing the price of all VMs of different types used in the workflow execution. The price of each VM is calculated based on its type and time duration. The time duration (completion time) is calculated based on the number of hours a particular VM executes from its instantiation to termination. Moreover, to execute the entire workflow, multiple VMs of different types can be used. The total computational cost of SWFA can be calculated as follows (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013):

$$TCC = \sum [Acual\ endTime - Acual\ startTime] \times prices\ of\ utilised\ VMs \quad (4.10)$$

4.1.3.2 Assumptions of SWFS Cost Optimisation Model

In order to correctly perform the functionalities of the SWFS cost optimisation model, the following are the main assumptions that have been considered in the proposed approach:

- The task dependencies of the considered SWFA should be modeled in DAG.
- The average computational time of a task running on a particular virtual machine should be known prior (historical data) to start the scheduling process.
- The data communication cost between tasks if scheduled on different sites is assumed to be constant.
- Once a task is running on a VM, then that VM cannot be occupied by any other task.
- The task executions of a given SWFA are assumed to be non-preemptive.

4.2 Dynamic Hyper-Heuristic Algorithm (DHHA)

The Dynamic Hyper-Heuristic Algorithm (DHHA) is the main part of the proposed Completion Time Driven Hyper-Heuristic (CTDHH) approach. A hyper-heuristic is a search methodology or learning mechanism for selecting existing heuristics in order to solve computationally intensive problems. More details about the comparison between the heuristic approaches can be found in Section 2.4.1. A hyper-heuristic operates on a large search space using a set of heuristic or meta-heuristic components rather than independently finding solutions. The originality of a hyper-heuristic algorithm is related to Operation Research in terms of machine learning method, which is utilized to find optimal or near

optimal solutions for computational-intensive search problems (i.e. real-world problems of NP-complete type) (Burke, Kendall, et al., 2005). Usually a hyper-heuristic algorithm considers selecting a list of different heuristic components (or heuristic algorithms) to efficiently optimise the given problem. As shown in Figure 4.7, the hyper-heuristic algorithm contains two separated levels, where the first level is High Level Heuristic (HLH) (i.e. the strategy) and the second level consists of the employed Low Level Heuristic (LLH) algorithms. In this study, the employed LLH algorithms of the proposed hyper-heuristic algorithm are considered as perturbative meta-heuristic, which operates on complete solutions. DHHA employs four well-known population-based meta-heuristic algorithms including, genetic algorithm, particle swarm optimisation, invasive weed optimisation, and hybrid invasive weed optimisation. The employed algorithms act as LLH algorithms in this research context. To provide a high level of abstraction to the hyper-heuristic algorithm, it has been assumed that there is a barrier (i.e. domain barrier) between the HLH and LLH (Glover & Kochenberger, 2006). The domain barrier is utilized to allow the flow of cost optimisation criteria from the LLH to HLH, which helps in evaluating the performance of candidate solutions generated by the employed meta-heuristic LLH algorithms.

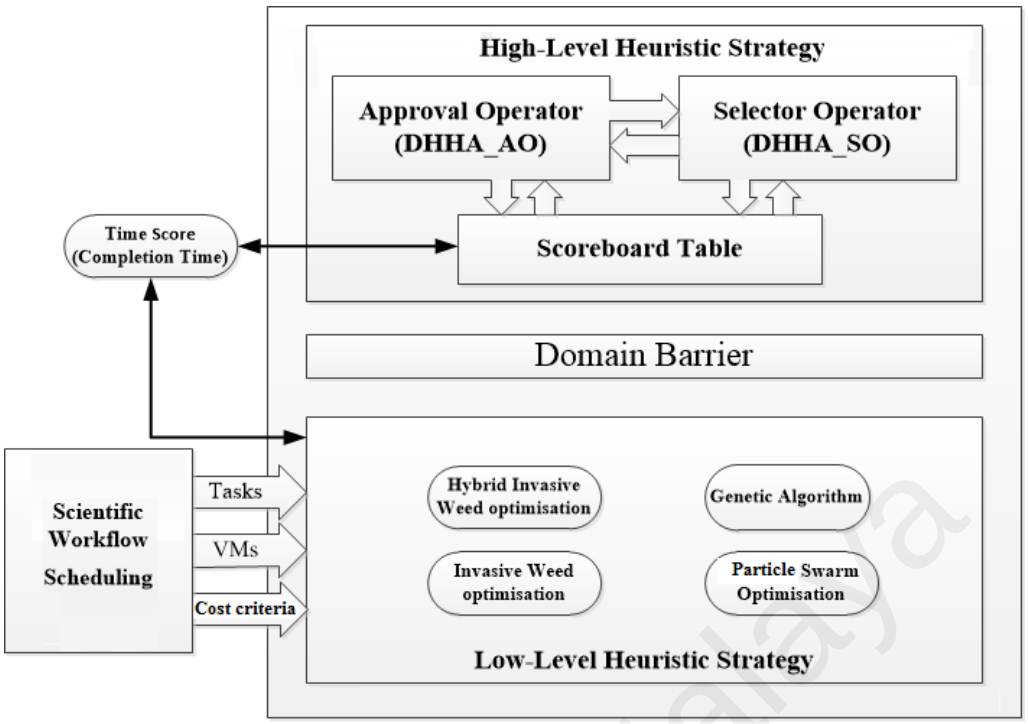

Figure 4.7: Dynamic Hyper-Heuristic Algorithm (DHHA)

The main purpose of HLH strategy is to intelligently guide the search process based on the performance of the employed meta-heuristic LLH algorithms (Burke et al., 2009; P. Cowling, Kendall, & Soubeiga, 2000). The proposed dynamic hyper-heuristic algorithm adopts a novel online learning mechanism, which is based on the feedback from the search space (i.e. by applying a LLH algorithms) while solving the targeted optimisation problem. This learning mechanism is far better than the existing hyper-heuristic algorithms that use a random LLH algorithm selection mechanism, since feedback provides a complete detail of each LLH algorithm. Moreover, the online learning mechanism of the proposed hyper-heuristic algorithm can dynamically guide the search process' decisions by selecting suitable employed LLH algorithms. Several initial solutions are generated by the employed meta-heuristic LLH algorithms and the HLH strategy decides which of these LLH algorithms will be selected for the next iteration. The HLH strategy employs the ***Selector Operator (DHHA_SO)*** and ***Approval Operator (DHHA_AO)*** to dynamically select the

most suitable candidate meta-heuristic algorithm from the employed LLH algorithms; thus, it ultimately helps to successfully make a good use of the employed LLH algorithms (Hussin, 2005). The HLH strategy decision can be done using the ***DHHA_SO*** to choose the most suitable meta-heuristic LLH algorithm based on the computed ***Time Score (TS)***, which ultimately is applied on the current solution. After that the ***DHHA_AO*** will decide whether to employ the same LLH algorithm again or employ another LLH algorithm. Thus, the newly defined operators can enhance the learning ability of the HLH strategy for each iteration of the proposed algorithm.

The proposed dynamic hyper-heuristic approach uses several successive stages, starting from generating the initial solutions (i.e. beginning stage) till the termination criteria (i.e. last stage). More details of each stage of the proposed dynamic hyper-heuristic algorithm are discussed in the following sections. Note that each of these stages contains a number of steps. Algorithm 1 shows the main steps of the proposed dynamic hyper-heuristic algorithm.

4.2.1 Stage 1: Initial Stage (Steps 1-7)

As it has been mentioned in the cost optimisation model of SWFS, the main three inputs to the proposed algorithms are: (i) list of SWFA tasks (Section 4.1.1), (ii) list of VMs (Section 4.1.2), and (iii) the cost optimisation criteria (Section 4.1.3). As shown in Algorithm 1, the first step of the initial stage of the proposed DHHA is running each of the four employed meta-heuristic algorithms (i.e. genetic algorithm, particle swarm optimisation, invasive weed optimisation, and hybrid invasive weed optimisation). In other words, the proposed algorithm run each ***(h)*** of the employed LLH algorithms ***(H)*** to schedule the submitted SWFA tasks based on the available VMs for five times for each particular scenario. The underlying reason of running each of the LLH algorithms for five times is that the proposed algorithm does not take long initialization time, which might ultimately

Algorithm 1 Dynamic Hyper-Heuristic Algorithm

Input: $DHHA\_SO(H)$ and $DHHA\_AO(H)$ where H is the set of the employed Low Level Heuristic algorithms
{The input for H are: Set of workflow tasks, Set of available VMs, Cost optimisation criteria}

Output: The most optimal solution for cost optimisation of SWFS

```
Initialization
for 1 to 5 do
    Run h, ∀h ∈ H
end for
Get the AV_M for the 5 runs
{Where AV_M representing the TS value}
for 1 to 30 do
    Sort the TS based on the completion time of h
    {from the lowest to the highest based on the Scoreboard table}
    loop
        Apply DHHA_SO(h), ∀ h ∈ H
        Choose the h with the lowest ST from the scoreboard table
        {Based on the smallest TS value in scoreboard table}
        Execute the chosen h
        Apply DHHA_AO(h), ∀ h ∈ H
        Examine the results based on the DHHA_AO(h) formula
        Approve h based on the results
        Update TS order in "Scoreboard Table" and update no. of runs in "Run Table"
    end loop
end for
Terminate the algorithm
{Till termination criteria are satisfied, which is the maximum number of runs = 30}
```

affect the overall completion time of the whole processes of the proposed algorithm. At each run of the meta-heuristic algorithms, the proposed DHHA considers three scheduling stages, which are considered as the LLH of the proposed algorithm. Section 4.1.3 provides the details of all the scheduling stages along with the cost optimisation criteria that have been considered for scheduling each of the LLH algorithms.

As previously mentioned, there are a different number of tasks (formula 4.2) and a different number of VMs (formula 4.5) for each type of the considered scenarios. It is worth to mention that the number of submitted workflow tasks is directly affected by the size of SWFA datasets. In other words, if the number of submitted SWFA tasks increases, then the WfMS will be required to compute a larger size of datasets, which ultimately increases

Table 4.3: Example of the nine considered scenarios

Scenario	No. of Tasks	No. of VMs
1	30	2
2	30	4
3	30	8
4	100	2
5	100	4
6	100	8
7	1000	2
8	1000	4
9	1000	8

the overall completion time. Thus, the number of workflow tasks is affected by the type of considered SWFA, which ultimately affects the dependencies and complexity between the workflow tasks. In this research, to effectively evaluate the computational-intensiveness and data-intensiveness of the proposed algorithm, nine different types of scenarios have been considered, where each scenario contains a different number of workflow tasks and a different number of VMs. Table 4.3 shows an example of the information (i.e. number of tasks and number of VMs) for the nine considered scenarios.

After successfully running each of the employed LLH algorithms for five times for all the considered scenarios, the next step of this stage is calculating the completion time (makespan) values ***(M)*** based on formula 4.9. The underlying intention about the completion time calculation is to provide information about the performance of the employed LLH algorithms in order to offer an online learning mechanism. Thus, the computed completion time directs the HLH strategy during the searching processes for the most optimised scheduling solution by selecting the most suitable LLH for the next run. The next step is to calculate the ***Average Value (AV_M)*** of completion time for all employed LLH algorithms for five runs of the targeted scenarios. The results of the average completion time value of each scenario will be considered as the computed ***Time Score (TS)***, which will be stored in a database table called as "scoreboard table". The time score

attribute of the scoreboard table is sorted based on the value of average completion time for each meta-heuristic algorithm and for each scenario. In this way, the best time score value is computed, which is always the overall minimum completion time value.

4.2.2 Stage 2: Selection Stage (Steps 8-11)

The ***DHHA_SO*** of the HLH strategy are based on the computed ***TS*** of the employed meta-heuristic LLH algorithms. As shown in Algorithm 1, the ***DHHA_SO*** operator chooses the most optimal computed time score for the considered scenarios, which depends on the number of available VMs and size of SWFA datasets. At each iteration, DHHA runs the selected LLH to schedule the submitted workflow tasks based on the considered scenario. The performances of the attained time score will ultimately be used by ***DHHA_SO*** to determine which LLH algorithm will be selected for the next iteration. The scoreboard table contains the time score and scenario details of each run of the employed meta-heuristic LLHs (i.e. algorithm name, problem size, available VMs, time score). In this way, at each run the proposed DHHA algorithm usea the high-level ***DHHA_SO*** to choose the LLH algorithm with the ***Lowest Time Score*** ($TS_{Lowest}(h)$) among other employed LLHs ***(H)*** for that specific scenario. Eq. 4.11 represents the defined ***DHHA_SO*** of the proposed algorithm.

$$DHHA\_SO(h) = TS_{Lowest}(h), TS_{Lowest}(h) \in TS_{List}(h) \tag{4.11}$$

The next step in the selection stage is to execute the chosen LLH ***(h)*** by following three scheduling stages that have been described in detail in Section 4.1.3 along with the cost optimisation criteria in Section 4.2.3. Once the selected LLH(h) is successfully executed, the proposed algorithm automatically updates ***TS*** value based on the calculated completion

time value ***TS(h)***.

For example, for scenario 4, if the IWO meta-heuristic algorithm (i.e. one of the employed LLH algorithm) attained the lowest average completion time comparing with other employed LLH algorithms (i.e. Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Hybrid Invasive Weed Optimisation (HIWO)), then the average completion time of IWO will be at the top of ***TS(h)*** attribute of the scoreboard table, and IWO will be chosen by ***DHHA_SO(h)*** for the next iteration.

4.2.3 Stage 3: Approval Stage (Steps 12-14)

As shown in Algorithm 1, for each single run, the ***DHHA_AO*** accepts the LLH algorithm that has the lowest computed time score, and updates the scoreboard and "run" table. The run table contains the run details of each run (i.e. number of runs) of an employed LLH algorithm. The proposed dynamic hyper-heuristic algorithm continuously updates the scoreboard table by replacing the existing time score with the lowest computed time score, which ultimately affects the total computational cost value for that run. Based on the scoreboard table, the proposed DHHA selects the most appropriate LLH algorithm for the next run.

After successfully running the LLH algorithm, the proposed DHHA compares the new completion time value with the best time score using ***DHHA_AO***. ***DHHA_AO*** is represented by threshold T value, which is based on two values, including ***Existing Time Score (ETS)*** and ***New Time Score (NTS)***. ETS is the current ***TS*** value in the scoreboard table and ***NTS*** is the newly achieved completion time after successfully running the LLH algorithm.

Based on comparison results between ***ETS*** and ***NTS*** values, the ***DHHA_AO*** can decide the desired ***TS*** value. It is worth to mention that, the original idea of calculating the ***TS*** value was by adding the ***ETS*** value to ***NTS*** value and then divide the result by constant

number (i.e., 2) to get the average. Thus, based on conducted experiments that has been done, it has been found that if the value of ***ETS*** or ***NTS*** multiplied by a constant number (i.e., 3) and then divide the result by a constant number (i.e., 4), then would help to improve the average value of ***TS***. Ultimately, this can also assist the proposed DHHA algorithm to choose the most appropriate LLH algorithm for the next iteration. Eq. 4.12 represents the ***DHHA_AO*** of the proposed algorithm. Choosing the constant number (i.e., 3) and constant number (i.e., 4) in Eq. 4.12, it has been derived from the conducted experiments.

$$ApprovalOperator(DHHA\_AO) = \begin{cases} NTS(h) \leq ETS(h)\, then\, TS = \frac{(ETS(h)*3)+NTS(h)}{4} \\ NTS(h) > ETS(h)\, then\, TS = \frac{ETS(h)+(NTS(h)*3)}{4} \end{cases} \tag{4.12}$$

It can be observed from Eq. 4.12 that, if the new completion time (***NTS***) value is less than or equal the existing time score (***ETS***) value, then the updated time score gives more value to ***ETS***, by multiply ***ETS*** with constant number (i.e., 3) constant number (i.e., 3) and then divide it by constant number (i.e., 4) to get the ***TS*** value.

In contrast, if the new completion time (***NTS***) value is greater than the existing time score (***ETS***) value then the updated time score gives more value to ***NTS*** by multiply ***NTS*** with constant number (i.e., 3) and then divide it by constant number (i.e., 4) to get the ***TS*** value. In this way, the proposed DHHA algorithm always gives compliment to the best value of ***ETS*** and ***NTS*** and at the same time, it does not affect too much the new ***TS*** value for the next iteration.

The purpose of multiplying completion time value by three and divided it by four is to prevent having big change in time score value, which would ultimately affect the selection criteria of the high-level selector. Additionally, the multiplication by three is based on

Table 4.4: Time score result from experimentation in the real-world environment

Problem Size (Degree)	**1**	**1**	**1**	**2**	**2**	**2**
Available VMs	**2**	**4**	**8**	**2**	**4**	**8**
DHHA_AO=*1/1	375	349	353	1066	1027	1012
DHHA_AO=*5/6	373	346	352	1064	1023	1002
DHHA_AO=*3/4	370	345	350	1053	997	963

the experiments that have been conducted in the experimentation stage using different values instead of three. Table 4.4 shows the experimentation result from the real-world environment evaluation. Thus, three values always show the best value. As it can be seen than, the ***TS*** result of the defined approval operation is the most optimal value comparing with other values. So, on the next run, the high-level selector chooses the LLH algorithm based on the updated time score of that specific scenario.

Based on the explanation of Eq. 4.12 and from Table 4.4, it can be seen that when the value of ***ETS*** or ***NTS*** multiplied by a constant number (i.e., 1) and then divide the result by a constant number (i.e., 1), in ***DHHA_AO*** (i.e., **DHHA_AO=*1/1**, the ***TS*** result is slightly affected by causing a large differences in ***TS*** value, which ultimately affects the approval discussion to choose the LLH for the next run.

At the same time, when the value of ***ETS*** or ***NTS*** multiplied by a constant number (i.e., 5) and then divide the result by a constant number (i.e., 6), is considered in ***DHHA_AO*** (i.e., **DHHA_AO=*5/6**, the ***TS*** result does not improve that much by causing a very small differences in ***TS*** value, which ultimately affects the approval discussion to choose the LLH for the next run.

Thus, based on the conducted experiments, it can be concluded that it is very important to use the most appropriate ***DHHA_AO*** value by multiplying ***ETS*** or ***NTS*** values by a constant number (i.e., 3) and then divide the result by a constant number (i.e., 4), (i.e., **DHHA_AO=*3/4**) to achieve the most optimal ***TS*** value, as this value will affect the approval discussion to choose the LLH for the next run.

With this dynamic LLH algorithm selection mechanism, the proposed hyper-heuristic algorithm helps towards finding a more optimal solution for the cost optimisation problem of SWFS in cloud environment. It is worth to mention that due to the domain barrier, the HLH strategy does not have enough knowledge about the nature of SWFS problem, while the ***DHHA_SO*** and ***DHHA_AO*** translate the scheduling problem (that has been defined in Section 4.1) as the problem domain by adopting the computed ***TS*** to choose the most appropriate algorithm from the employed LLH algorithms.

4.2.4 Stage 4: Termination Stage (Step 15)

The steps in selection and approval stages are repeated until the stop criteria is met. Note that the considered stop criteria of the proposed algorithm are the maximum number of runs. Based on the literature review setting (Talukder et al., 2009, 2007; J. Yu, Kirley, & Buyya, 2007), the maximum number of runs has been fixed to 30 as the stop criteria. Finally, the result of all successful runs has been collected and analysed, and also compared them with the baseline meta-heuristic algorithms.

4.2.5 Example of the Proposed Algorithm

In this section, an example of the proposed dynamic hyper-heuristic algorithm is provided. The given example is about the considered scenario, where the size of SWFA tasks is the smallest problem size and number of available VMs is 8.

At the first step of the proposed DHHA (Figure 4.8), all the employed meta-heuristic LLH algorithms (i.e. genetic algorithm, particle swarm optimisation, invasive weed optimisation, and hybrid invasive weed optimisation) run for five times. Then, in the second step, average completion time ***(AV_M)*** from the results of running all LLH algorithms are taken and stored in a scoreboard table.

Scoreboard Table

LLH	Time Score	No. of Runs	Problem size	Available VMs
GA	380.5	5	Smallest	8
PSO	378.9	5	Smallest	8
IWO	373.5	5	Smallest	8
HIWO	392.1	5	Smallest	8

Figure 4.8: First step of the proposed DHHA

The proposed approach sorts the obtained average completion time (TS), where the TS value is arranged from the smallest to the largest value (Figure 4.9).

Scoreboard Table

LLH	Time Score	No. of Runs	Problem size	Available VMs
IWO	373.5	5	Smallest	8
PSO	378.9	5	Smallest	8
GA	380.5	5	Smallest	8
HIWO	392.1	5	Smallest	8

Figure 4.9: Sorting the scoreboard table based on the TS value

As shown in Figure 4.10, after applying the ***DHHA_SO*** the HLH strategy chooses the LLH algorithm ***(h)*** with the lowest ***TS***. Based on the afore-mentioned online learning mechanism, the IWO algorithm has shown to have the smallest ***TS*** value. Therefore, IWO is used on the next run to schedule the SWFA for this scenario.

Scoreboard Table

LLH	Time Score	No. of Runs	Problem size	Available VMs
IWO	373.5	5	Smallest	8
PSO	378.9	5	Smallest	8
GA	380.5	5	Smallest	8
HIWO	392.1	5	Smallest	8

Figure 4.10: Apply the Selector Operator (DHHA_SO)

Figure 4.11 shows the result after applying ***DHHA_SO*** and ***DHHA_AO***. Next, the scoreboard table and the run table are updated based on the lowest ***TS*** value (i.e., 362.15), which achieved by IWO algorithm. Thus, after each run the proposed algorithm updates the scoreboard table and run table based on the sorted the ***TS*** value.

Scoreboard Table

LLH	Time Score	No. of Runs	Problem size	Available VMs
IWO	362.15	6	Smallest	8
PSO	378.9	5	Smallest	8
GA	380.5	5	Smallest	8
HIWO	392.1	5	Smallest	8

Run Table

Run #	LLH	New TS
1	IWO	**350.8**
.	.	.
.	.	.
30	GA	**379.2**

Figure 4.11: Updating scoreboard table and run table

The above-mentioned steps are repeated for 30 times by applying ***DHHA_SO*** and ***DHHA_AO*** (Figure 4.12). The termination criteria of the proposed DHHA is the maximum number of runs (i.e. 30 times).

Scoreboard Table

LLH	Time Score	No. of Runs	Problem size	Available VMs
IWO	362.15	20	Smallest	8
PSO	378.9	15	Smallest	8
GA	380.5	8	Smallest	8
HIWO	392.1	7	Smallest	8

Run Table

Run #	LLH	New TS
1	IWO	**350.8**
.	.	.
.	.	.
30	GA	**379.2**

Figure 4.12: Repeating the proposed operators of the HLH strategy till reaching termination criteria

4.3 Summary

The hyper-heuristic approach is an emerging class of meta-heuristic algorithms, which is integrated in such a manner that allows utilising the maximum strengths of the employed meta-heuristic algorithms to obtain an optimal solution. In the literature, the hyper-heuristic mechanisms achieved better performance in terms of shorter execution time compared to other optimisation mechanisms. Additionally, hyper meta-heuristic is a class of new advanced techniques that are capable of accelerating the run-time of a single meta-heuristic algorithm. The proposed DHHA employs four well-known population-based meta-heuristic algorithms, which act as Low Level Heuristic (LLH) algorithms (i.e. genetic algorithm, particle swarm optimisation, invasive weed optimisation, and hybrid invasive weed optimisation). In addition, the proposed algorithm enhances the native random selection way of existing hyper-heuristic approaches by incorporating the best computed workflow completion time that acts as a high-level selector to pick a suitable algorithm from the pool of LLH algorithms after each run. The HLH strategy of the Selector Operator (DHHA_SO) and Approval Operator (DHHA_AO) is depending on the completion time performance of the employed LLH algorithms.

The main aim of the proposed approach is to reduce the completion time and total computational cost to execute the SWFA. Based on the lowest achieved completion time,

the proposed approach dynamically guides the searching processes to find an optimal solution by continuously sorting the computed time scores (i.e. completion times of previous runs) of all LLH algorithms for each considered scenario and after every run. Consequently, the mechanism of the proposed completion time driven hyper-heuristic becomes more effective in a way allowing reusing and utilising the maximum strengths of the employed LLH algorithms in searching for the optimal solution of the targeted cost optimisation problem.

CHAPTER 5: EVALUATION AND ANALYSIS USING SIMULATION ENVIRONMENT

In order to achieve the third research objective of this research, which is *"To evaluate and analyse the performance of cost optimisation parameters (i.e., completion time and total computational cost) of the proposed approach"*, and based on the third stage of research methodology, in this chapter, all the relevant steps have been identified and discussed in order to evaluate the proposed completion time driven hyper-heuristic approach for cost optimisation of SWFS using simulation environment. The benefit of using the simulation environment is to investigate the performance of proposed approach based on different types of SWFAs. The proposed CTDHH approach has extensively evaluated by comparing it with four well-known types of meta-heuristics baseline algorithms (i.e., genetic algorithm, particle swarm optimisation, invasive weed optimisation, and hybrid invasive weed optimisation) as well as by comparing the proposed CTDHH approach with an existing hyper-heuristic approach named Hyper-Heuristic Scheduling Algorithm (HHSA). Appendix A provides a complete description of baseline approaches (i.e., GA, PSO, IWO, and HIWO) and the HHSA approach. Four types of SWFAs datasets (i.e., Epigenomics, Inspiral, Montage, SIPHT) have been used in the simulation experimentation, and for each of these SWFAs, there are three sizes which have been considered in order to evaluate the data intensiveness and computational intensiveness of the proposed approach. The experimentation setup of the simulation environment has been discussed. Finally, different types of statistical analysis have been conducted for the collected results from running the experimentation. A complete detail on the evaluation of the proposed approach in the simulation environment will be provided in the following sections.

5.1 Simulation Environment

The evaluation of any approach using the simulation environment would allow the researchers to perform simulations with different application configurations under controlled conditions. Also, based on the comparison of existing experimental tools (Section 3.3.1) from the state-of-the-art cost optimisation of SWFS approaches, it has been found that there are several types of simulations tools that have been utilised by the other researchers for evaluating the cost optimisation of SWFS approaches in cloud computing (i.e., CloudSim (Calheiros, Ranjan, Beloglazov, De Rose, & Buyya, 2011), GridSim (Buyya & Murshed, 2002), EMUSIM (Calheiros, Netto, De Rose, & Buyya, 2013), and CloudAnalyst (Wickremasinghe, Calheiros, & Buyya, n.d.). However, only the WorkflowSim (W. Chen & Deelman, n.d.) simulator is considered as the standard workflow execution model such as workflow mapper, workflow engine, workflow scheduler, clustering engine, provenance collector, workflow partitioner. WorkflowSim simulator is an open-access programming tools for developing and simulating WfMS implemented in a parallel and distributed environment (W. Chen & Deelman, n.d.).

The following are the main characteristics of WorkflowSim models:

- *Workflow Mapper* to map abstract workflows to concrete workflows that are dependent on execution sites.
- *Workflow Engine* handles the data dependencies.
- *Workflow Scheduler* matches jobs to the resources.
- *Clustering Engine* merges small tasks into a large job.
- *Provenance Collector* tracks the history of task/job execution.
- *Workflow Partitioner* divides the user workflow into multiple sub-workflows.

Implementation of the proposed CTDHH approach has been done on WorkflowSim simulator by setting it up on Eclipse editor. Next, the WorkflowSim simulator has been

utilised to develop and evaluate the proposed approach by comparing it with four meta-heuristics baseline algorithms as well as by comparing the proposed CTDHH approach with the HHSA approach. WorkflowSim simulator uses the data collected from actual scientific workflow executions to generate synthetic workflows resembling those used by real-world scientific applications. The scheduling-based simulation allows in clearly understanding and assessing the scheduling process and to easily determine various types of scenarios that depend on the number of available VMs and size of SWFA dataset to completely investigate the performance of the proposed approach. The WorkflowSim simulator includes several types of examples and each of these examples is used for different kinds of challenges of workflow management processes (e.g., networking, clustering, power estimation, cost computation, scheduling). Additionally, the WorkflowSim simulator has the required cloud CloudSim (Calheiros et al., 2011) techniques that support the execution of workloads in cloud computing infrastructures and services. CloudSim classes and packages have been used to read the execution time value of tasks for the workload traces.

5.2 Scientific Workflow Applications

As already defined the SWFA in the previous chapters, the SWFA consists of multiple tasks, which are necessary to complete a submitted workflow (Wieczorek et al., 2009; Ma et al., 2009; Malawski et al., 2014; Pacini, Mateos, & Garino, 2014). The components of these tasks can be any executable instances (e.g., load sets, report sets, programs, and data). The other relation in a scientific workflow application is the relationship between the tasks/jobs and their data dependencies. As illustrated in Figure 5.1, there are five main types of relationships between the jobs and data of SWFAs, which are process, pipeline, data distribution, data aggregation and data redistribution.

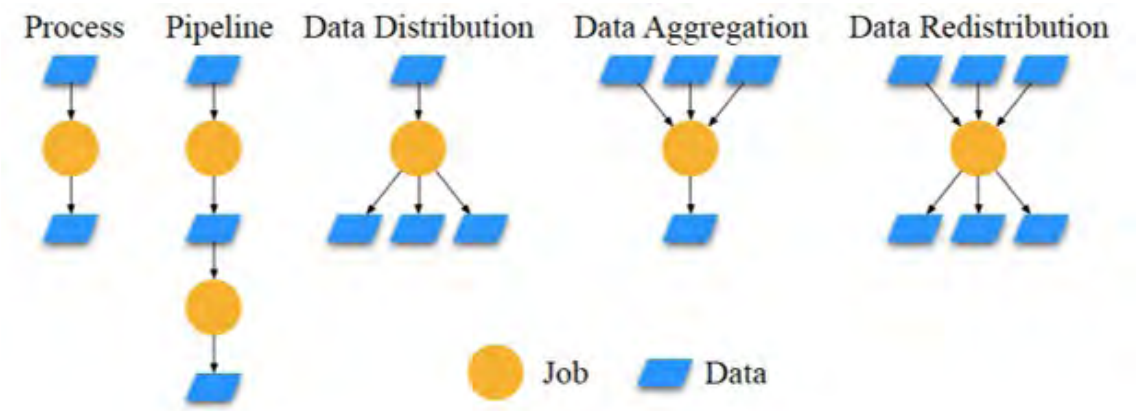

Figure 5.1: Types of the relationship between job and data of SWFAs

Additionally, in the clustering stage of WorkflowSim simulator, the tasks that have the same type of process can be represented as a job (Figure 5.1). In this way, similar tasks can be executed one time instead of repeating the same execution for several times.

Based on the literature review, four kinds of SWFAs (i.e., Epigenomics, Inspiral, Montage, SIPHT) have been considered as datasets to compare the proposed approach with the baseline approaches as well as to compare the proposed approach with HHSA approach. These SWFAs have been widely adopted by several researchers as standard SWFAs (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013; Malawski et al., 2012, 2014; Szabo et al., 2014). Each of the considered SWFAs has different computation-intensive tasks dependencies because it has been used to facilitate different scientific application. In order to evaluate the data-intensiveness of the proposed CTDHH approach, different sizes for each of the SWFAs (i.e., small, medium, and large) have been considered. Complete definitions for each of the considered SWFAs can be found in Appendix B. Table 5.1 represents the settings of the SWFAs that have been used to evaluate the proposed CTDHH approach based on the size of the application, number of tasks and number of edges.

Table 5.1: Specification of SWFA datasets

Dataset Name	Workflow Application	Number of Tasks	Number of Edges
CyberShake	Earthquake science	50	88
		100	180
		1000	1988
Epigenome	Biology	46	54
		100	122
		997	1234
LIGO Inspiral	Gravitational physics	50	60
		100	119
		1000	1233
Montage	Astronomy	50	106
		100	233
		1000	2485
SIPHT	Biology	60	66
		100	109
		1000	1096

5.3 Experimentation Setting

For running the simulation experiments, PC with the following specifications has been used: (i) operating system: Windows 10 Pro (64 bit), (ii) processor: Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, and (iii) Memory (RAM): 12.0 GB. Due to the reason that the WorkflowSim is programmed using Java language, Eclipse as integrated development environment has been used to run the WorkflowSim codes and to implement the proposed CTDHH approach. The code of WorkflowSim simulator (Version 1.0) has been directly imported from the GitHub website (https://github.com/WorkflowSim/WorkflowSim-1.0.git). After doing the necessary modifications to the directories of WorkflowSim code, existing provided examples of the simulator are successfully managed to run. It is worth to mention that the prices of VM is based on EC2 pricing list. In this research, the type of available computational resources (i.e., VMs) has been selected based on the comprehensive study of the existing reproaches in Section 3.3.1 and based on WorkflowSim simulation environment, the considered VM instance is equivalent to t2.small instance of Amazon EC2 website

(*Amazon EC2 instance types*, 2015).

5.4 Results and discussion of Simulation Environment

The proposed Completion Time Driven Hyper-Heuristic (CTDHH) approach has been compared with four baseline meta-heuristic algorithms (i.e., Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Invasive Weed Optimisation (IWO), and Hybrid Invasive Weed Optimisation (HIWO)) and an existing hyper-heuristic approach named Hyper-Heuristic Scheduling Algorithm (HHSA). The complete details for each of the used population-based approaches and HHSA approach is provided in Appendix A.

Four types of SWFAs (i.e., Epigenomics, Inspiral, Montage and Sipht) have been considered in the simulation environment, which is considered as standard SWFAs in several previous studies (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013; Y. Yang et al., 2008; H. Liu et al., 2011). For each of these SWFAs, there are nine scenarios that have been utilized in the experiments and each of these scenario has different number of workflow tasks and different number of VMs. Two main types of statistical analysis have been used in this research to evaluate the performance of the proposed CTDHH approach including descriptive statistical analysis as well as normality testing and significant statistical analysis.

5.4.1 Descriptive Statistic Analysis

In order to extensively analyse the collected data from experimentation, four types of descriptive statistical analysis have been used (i.e., average, minimum, maximum, and standard deviation (S.D.)). The descriptive statistic analysis method has been conducted to compare the proposed CTDHH approach with the baseline approaches and HHSA approach based on the completion time and total computational cost parameters. As has been discussed in Section 4.1.3, the value of completion time is calculated based on the

Table 5.2: Specification of scenarios for Epigenomics SWFA in simulation environment

Scenario	No. of Tasks	No. of VMs
1	24	2
2	24	4
3	24	8
4	100	2
5	100	4
6	100	8
7	997	2
8	997	4
9	997	8

total required time to finish executing the submitted workflow tasks. Moreover, due to the static submitting criteria, the value of total computational cost is driven from the completion time multiplied by the number of VMs multiplied by the price of the VM. In the following sections, Section 5.4.1.1, Section 5.4.1.2, Section 5.4.1.3, and Section 5.4.1.4, the descriptive statistic analysis of experimentation results have been presented and discussed for each of the considered SWFAs (i.e., Epigenomics, Inspiral, Montage and Sipht).

5.4.1.1 Epigenomics-Genome Sequencing SWFAs

This section provides the descriptive statistic analysis result's discussion of Epigenomics SWFA. Table 5.2 shows the specification of the utilized scenarios for Epigenomics SWFA. Based on the number of tasks (which depends on the dataset size) and number of VMs, the first three scenarios are considered as the smallest datasets scenarios, while the last three scenarios are considered as the largest scenarios. Appendix B.1 describes the Epigenomics SWFA datasets in detail.

This section has been divided into two parts. The first part presents the comparison between proposed CTDHH approach and the baseline approaches (i.e., GA, PSO, IWO, HIWO), while the second part presents the comparison between proposed CTDHH

approach and HHSA approach.

A- Comparison between Proposed CTDHH Approach and Baseline Approaches

In this section, the completion time and total computational cost statistical results based on the comparison between proposed CTDHH approach and baseline approaches have been presented for each of the nine considered scenarios. Based on the reviewed related works, the average value has been considered as standard benchmarks by other researchers in this area of research (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013).

i. Completion time results:

Table 5.3 represents the descriptive statistical analysis for completion time results of Epigenomics SWFA for all considered nine scenarios.

Table 5.3: Completion time comparison between CTDHH and baselines for Epigenomics

Scenario-1					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	22.704	22.704	22.704	22.704	14.308
MIN	22.704	22.704	22.704	22.704	5.948
MAX	22.704	22.704	22.704	22.704	26.262
S.D.	7.2269E-15	7.2269E-15	7.2269E-15	7.2269E-15	5.537990448
Scenario-2					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	25.88666667	27.45866667	26.34533333	26.43466667	23.343
MIN	22.924	22.924	22.924	22.924	7
MAX	28.484	28.484	28.484	28.484	40.942
S.D.	2.625728054	1.552909912	2.153623624	1.782943969	7.201940557
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	33.75066667	35.444	33.852	33.00933333	24.0434
MIN	23.364	26.484	23.364	23.364	13.176
MAX	42.484	42.484	42.484	42.484	39.452
S.D.	6.956561444	5.965972476	6.873689167	7.881233814	7.224372716
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	44.672	44.0328	44.672	44.672	26.38806667
MIN	44.672	25.496	44.672	44.672	14.604
MAX	44.672	44.672	44.672	44.672	65.118
S.D.	2.16807E-14	3.501042588	2.16807E-14	2.16807E-14	12.50616675
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	53.19066667	59.04773333	58.72133333	58.38	33.1116
MIN	44.892	44.892	44.892	44.892	12
MAX	65.652	65.652	65.652	65.652	66.852
S.D.	8.789534425	8.722158033	6.301316192	8.247980066	11.12517507
Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	82.33733333	81.83773333	87.01733333	88.452	70.96763333
MIN	45.332	36.476	45.332	56.052	36.08
MAX	110.052	110.052	110.052	110.052	105.78
S.D.	19.98126221	20.96481981	20.77926747	19.11030774	16.13633771
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**

AVERAGE	127.126	124.9986	127.126	127.126	119.3176
MIN	127.126	101.85	127.126	127.126	109.54
MAX	127.126	127.126	127.126	127.126	133.356
S.D.	7.2269E-14	5.812743103	7.2269E-14	7.2269E-14	9.14656196
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	242.162	237.1503333	236.818	236.0023333	187.4326667
MIN	127.346	127.346	127.346	114.95	127.346
MAX	317.576	317.576	317.576	324.036	217.476
S.D.	65.50383503	83.41390572	79.65963722	73.17068109	43.21402736
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	586.776	425.3893333	449.8770333	570.026	359.4460667
MIN	218.276	217.476	109.316	218.276	318.776
MAX	720.776	720.776	720.776	720.776	419.276
S.D.	177.7014156	192.4171075	208.9468766	184.276252	44.27171072

From Table 5.3, it can be clearly seen that the average completion time of proposed CTDHH approach significantly outperformed the baseline approaches for all scenarios. This is due to the fact that the hyper-heuristic mechanism always chooses the selected LLH based on their completion time performance at each iteration. Moreover, due to the utilized simulation environment, the S.D. results are considered higher than those in the the real-world environment. However, the average value of the completion time is more important than the S.D. as completion time value is always considered as standard benchmark by other researchers in this area of research (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013).

Figure 5.2 illustrates the average completion time of CTDHH and baselines for Epigenomics SWFA based on the calculated results in Table 5.2. This figure has five charts and each of these charts is representing the relationship between the average completion time of each considered scenarios; where x-axis represents the considered scenarios, and y-axis denotes average completion time in seconds.

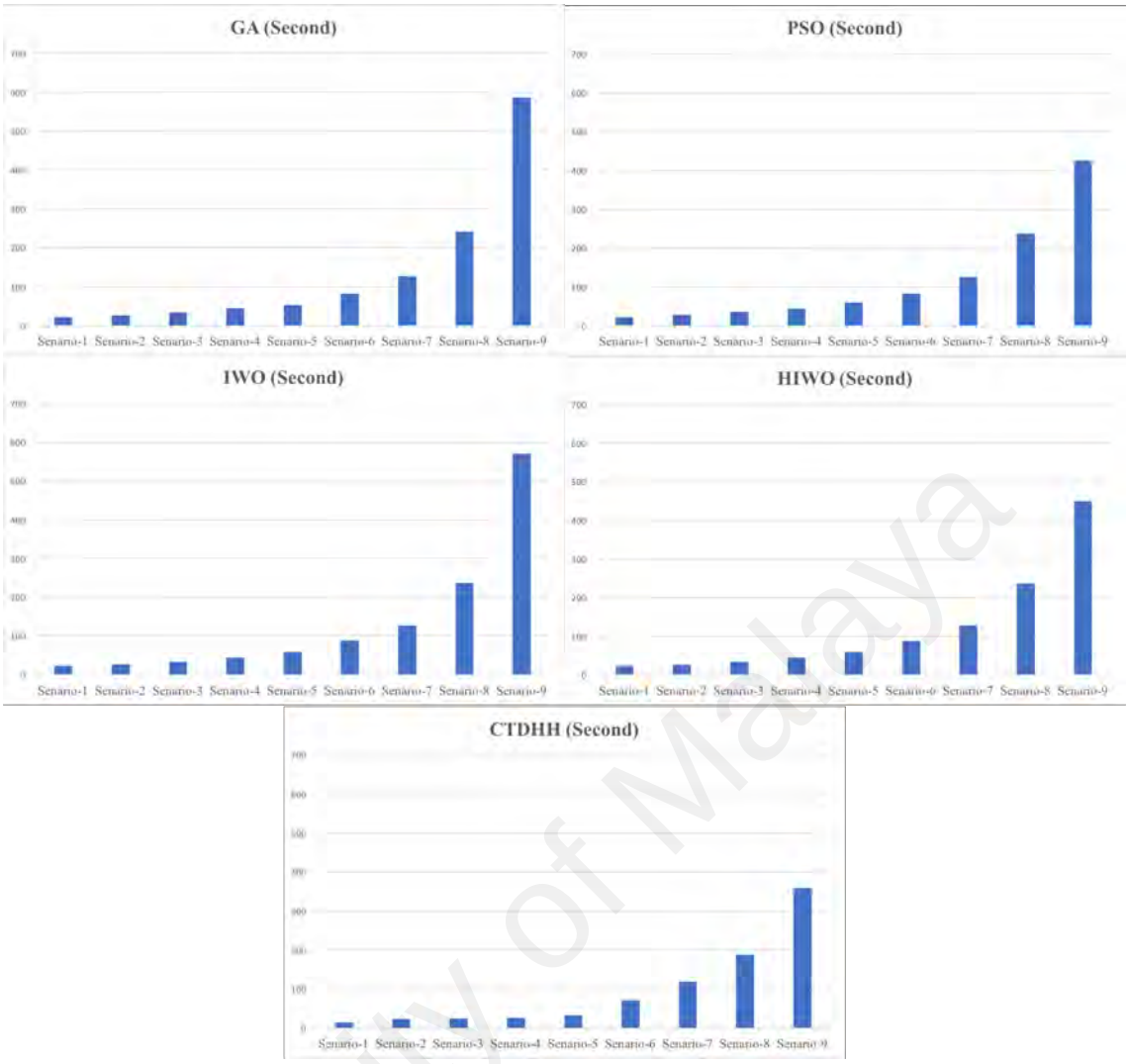

Figure 5.2: Average completion time of CTDHH and baselines for Epigenomics SWFA

It is evident from Figure 5.2 that the average completion time is increasing as the number of workflow tasks and number of available VMs are increased for all approaches. The average completion time of the proposed CTDHH approach outperforms other baseline approaches for all considered scenarios. The underlying fact of achieving this desired result is that the proposed CTDHH dynamically chooses the LLH using selection and approval operators. As a result, the proposed CTDHH approach attains minimal average completion time as compared to those of the employed baseline approaches.

Figure 5.3 shows the comparison between average completion time of all approaches

(i.e., proposed and baselines) for Epigenomics SWFA for all considered scenarios.

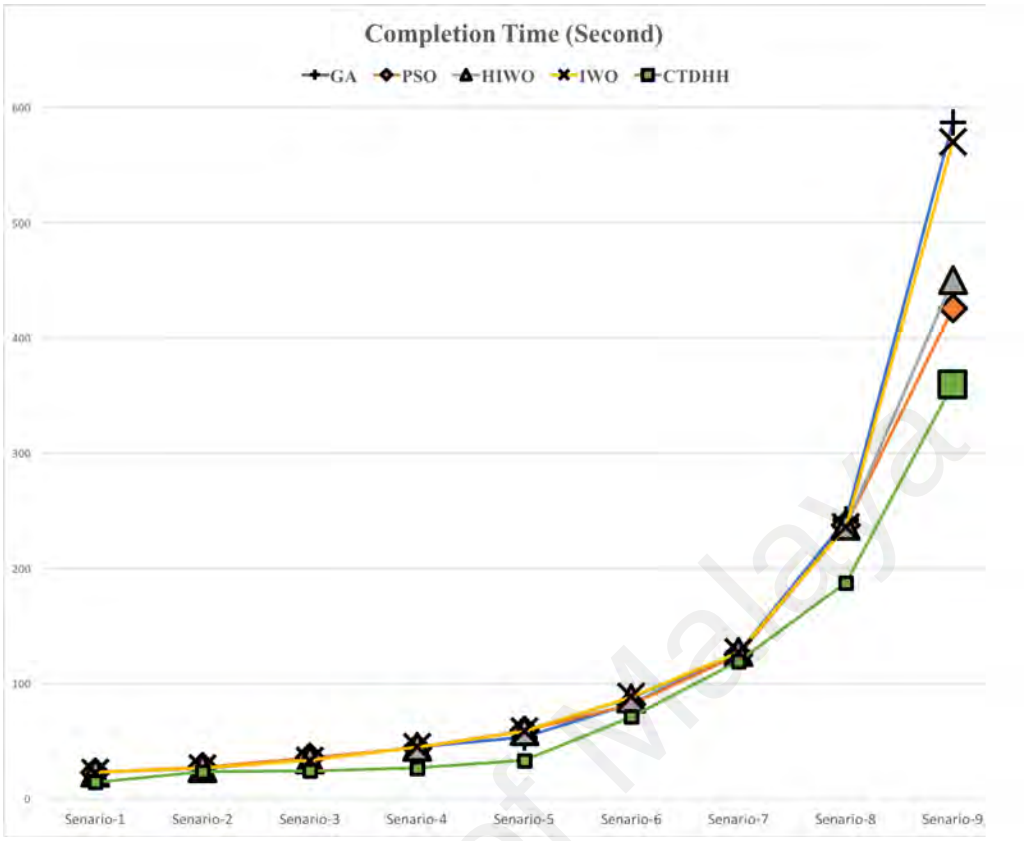

Figure 5.3: Average completion time of all approaches for Epigenomics SWFA

From Figure 5.3, it can be concluded that as the size of a dataset increases along with several available VMs, the proposed CTDHH approach attains better performance (in terms of completion time) than baseline approaches. Improved performance is not so obvious for the proposed approach until scenario 6, this is because of lesser a number of tasks (i.e., maximum up to 100). In contrast, as the number of tasks (i.e., 100 to 997) increases for scenarios 6 to 9, significant differences of average completion time are attained by the proposed approach over all other approaches.

ii. Total computational cost results:

Table 5.4 represents the descriptive statistical analysis results for total computational cost by comparing the proposed CTDHH approach and baseline approaches of Epigenomics SWFA for all considered scenarios.

Table 5.4: Total computational cost comparison CTDHH and baselines for Epigenomics

Scenario-1					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.0266454	0.0266454	0.0266454	0.0266454	0.0167919
MIN	0.0266454	0.0266454	0.0266454	0.0266454	0.0069806
MAX	0.0266454	0.0266454	0.0266454	0.0266454	0.0308211
S.D.	4.962E-18	4.962E-18	4.962E-18	4.962E-18	0.0064994
Scenario-2					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	0.0607612	0.064451	0.0618378	0.0620474	0.0547907
MIN	0.0538072	0.0538072	0.0538072	0.0538072	0.0164304
MAX	0.0668576	0.0668576	0.0668576	0.0668576	0.0960991
S.D.	0.0061631	0.003645	0.005055	0.0041849	0.0169044
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	0.1584391	0.1663883	0.1589148	0.154959	0.1128693
MIN	0.10968	0.1243265	0.10968	0.10968	0.0618534
MAX	0.1994369	0.1994369	0.1994369	0.1994369	0.1852035
S.D.	0.0326569	0.0280067	0.0322678	0.0369977	0.0339141
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	0.0524271	0.0516769	0.0524271	0.0524271	0.030969
MIN	0.0524271	0.0299221	0.0524271	0.0524271	0.0171393
MAX	0.0524271	0.0524271	0.0524271	0.0524271	0.0764225
S.D.	0	0.0041088	0	0	0.0146772
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVAERAGE	0.1248491	0.1385968	0.1378307	0.1370295	0.0777195
MIN	0.1053705	0.1053705	0.1053705	0.1053705	0.0281664
MAX	0.1540984	0.1540984	0.1540984	0.1540984	0.156915
S.D.	0.0206308	0.0204726	0.0147904	0.0193597	0.026113
Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.3865244	0.3841791	0.4084942	0.2631305	0.3331505
MIN	0.2128065	0.1712329	0.2128065	0.2631305	0.169374
MAX	0.5166281	0.5166281	0.5166281	0.5166281	0.4965736
S.D.	0.0938	0.0984173	0.0975462	0.0897114	0.0757504
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.1491951	0.1466984	0.1491951	0.1491951	0.1400311
MIN	0.1491951	0.1195312	0.1491951	0.1491951	0.1285561
MAX	0.1491951	0.1491951	0.1491951	0.1491951	0.1565066
S.D.	7.94E-17	0.0068218	7.94E-17	7.94E-17	0.0107344
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.5684026	0.5566393	0.5558592	0.5539447	0.439942
MIN	0.2989065	0.2989065	0.2989065	0.2698106	0.2989065
MAX	0.7454144	0.7454144	0.7454144	0.7605773	0.5104597
S.D.	0.1537506	0.1957891	0.1869771	0.1717462	0.101432
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	2.7545613	1.9969477	2.1119027	2.6759301	1.6873836
MIN	1.0246749	1.0209193	0.513173	1.0246749	1.4964621
MAX	3.3836109	3.3836109	3.3836109	3.3836109	1.9682493
S.D.	0.8342015	0.9032829	0.9808802	0.8650664	0.2078291

Table 5.4 highlights that the average total computational cost of the proposed CTDHH approach has the lowest average value for all scenarios comparing with other baseline approaches. This is mainly due to the fact that the average total completion cost value of

SWFS depends on the average completion time. This has confirmed that the proposed HLH strategy has successfully improved the performance of the proposed CTDHH approach by dynamically choosing the optimal solution for SWFS. However, in scenario 6, the computational cost value of IWO approach is slightly lower than the computational cost value of the proposed CTDHH approach and this due to fact that IWO approach has achieved shorter convergence time with complex SWFA (i.e., Epigenomics), which ultimately affected the total computational cost value.

Figure 5.4 illustrates the average total computational cost of the proposed CTDHH and baseline approaches for Epigenomics. It contains five charts and each of these charts is representing the relationship between the total computational cost ($/hour) and the considered scenarios.

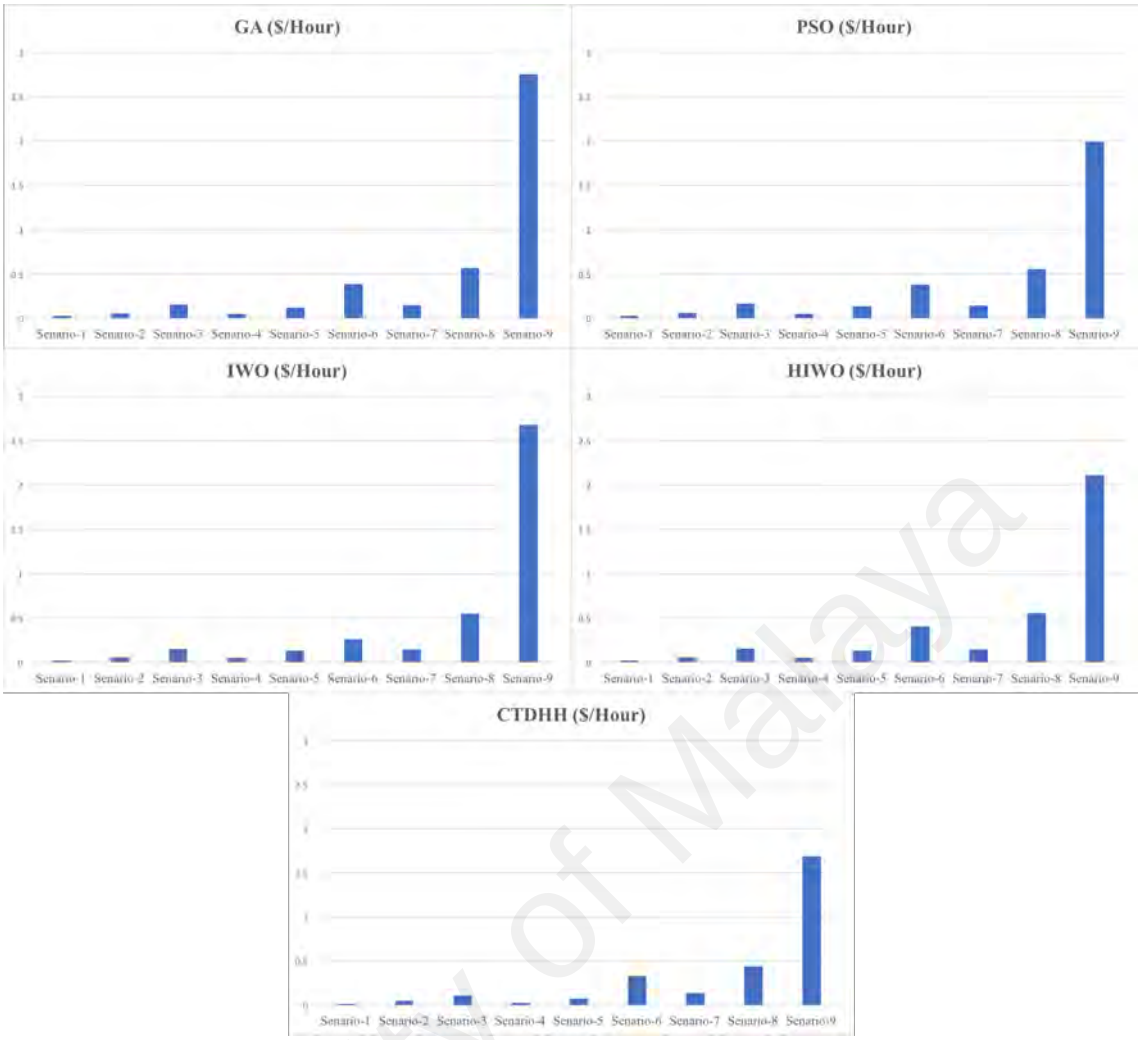

Figure 5.4: Average total computational cost of CTDHH and baselines for Epigenomics

Figure 5.4 shows that the total computational cost heavily depends on the completion time, number of available VMs as well as the number of workflow tasks for each of the considered scenarios. The proposed CTDHH approach always attains optimal performance results for most of the considered scenarios. In few scenarios, the proposed CTDHH approach could achieved longer convergence time with complex SWFA (i.e., Epigenomics), which ultimately affected the total computational cost value.

Figure 5.5 illustrates the comparison between average total computational cost ($/hour) values of all approaches for the considered scenarios.

Figure 5.5: Average total computational cost of all approaches for Epigenomics SWFA

It can be observed from Figure 5.5 that there is a significant difference between the performance of all approaches for scenario 6 to scenario 9, due to the limited searching space for the optimal solutions. This confirmed the fact that the total computational cost results heavily depend on the average completion time values, which are ultimately affected by the number of submitted SWFA tasks as well as the number of available VMs.

B- Comparison between Proposed CTDHH Approach and HHSA Approach

In this section, the descriptive statistical analysis has been presented for the completion time and total computational cost by comparing the proposed CTDHH approach with HHSA approach.

i. Completion time results:

Table 5.5 illustrates the descriptive statistical analysis of completion time by comparing

the proposed CTDHH approach and HHSA approach for Epigenomics for all considered scenarios.

Table 5.5: Completion time comparison between CTDHH and HHSA for Epigenomics

Scenario-1		
	HHSA	**CTDHH**
AVERAGE	17.4762	14.308
MIN	0.31	5.948
MAX	26.262	26.262
S.D.	6.595894153	5.537990448
Scenario-2		
	HHSA	**CTDHH**
AVAERAGE	25.88666667	23.343
MIN	22.924	7
MAX	28.484	40.942
S.D.	2.625728054	7.201940557
Scenario-3		
	HHSA	**CTDHH**
AVAERAGE	29.43326667	24.0434
MIN	19	13.176
MAX	46.096	39.452
S.D.	7.950334237	7.224372716
Scenario-4		
	HHSA	**CTDHH**
AVAERAGE	39.0698	26.38806667
MIN	27.196	14.604
MAX	46.534	65.118
S.D.	7.899832584	12.50616675
Scenario-5		
	HHSA	**CTDHH**
AVAERAGE	43.0406	33.1116
MIN	22.612	12
MAX	67.242	66.852
S.D.	13.88392496	11.12517507
Scenario-6		
	HHSA	**CTDHH**
AVERAGE	74.4286	70.96763333
MIN	24.032	36.08
MAX	110.052	105.78
S.D.	27.08556185	16.13633771
Scenario-7		
	HHSA	**CTDHH**
AVERAGE	122.2767333	119.3176
MIN	109.63	109.54
MAX	133.356	133.356
S.D.	9.164143867	9.14656196
Scenario-8		
	HHSA	**CTDHH**
AVERAGE	258.9094	187.4326667
MIN	139.986	127.346
MAX	306.26	217.476
S.D.	53.18472592	43.21402736
Scenario-9		
	HHSA	**CTDHH**
AVERAGE	482.7182	359.4460667
MIN	189.19	318.776
MAX	720.776	419.276
S.D.	198.648087	44.27171072

It can be observed from Table 5.5 that the average completion time value of the proposed CTDHH approach is more optimised than HHSA approach. This is because of the reason

that the dynamic mechanism of the proposed CTDHH has replaced the random mechanism in the HHSA approach. Generally speaking, the dynamic mechanism enables the proposed CTDHH approach to intelligently determine the optimal solution rather than randomly finding the solution as of HHSA approach.

Figure 5.6 illustrates the average completion time of CTDHH and HHSA approach for Epigenomics SWFA. Figure 5.6 has two charts, where each chart is representing the relationship between the completion time and the considered scenarios for each of the approaches.

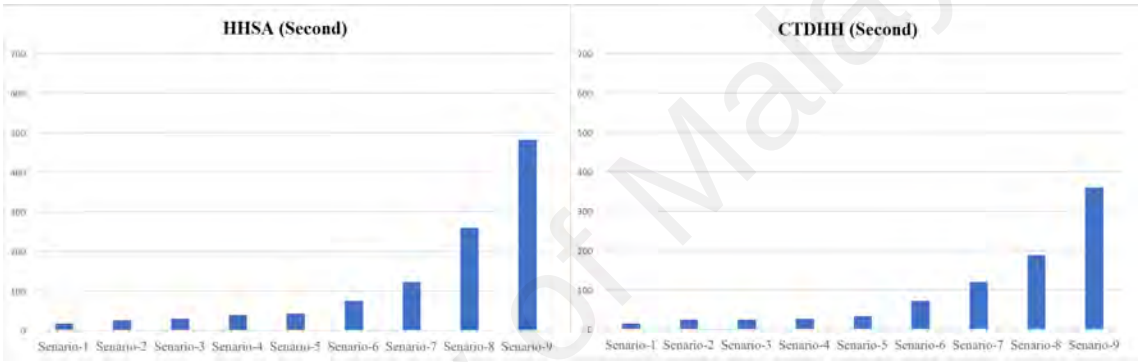

Figure 5.6: Average completion time of CTDHH and HHSA for Epigenomics SWFA

From Figure 5.6, it can be seen that the average completion time (in second) increases as the number of workflow tasks and number of available of VMs are increased for both approaches.

Figure 5.7 shows the average completion time of both approaches for Epigenomics SWFA of all the considered scenarios.

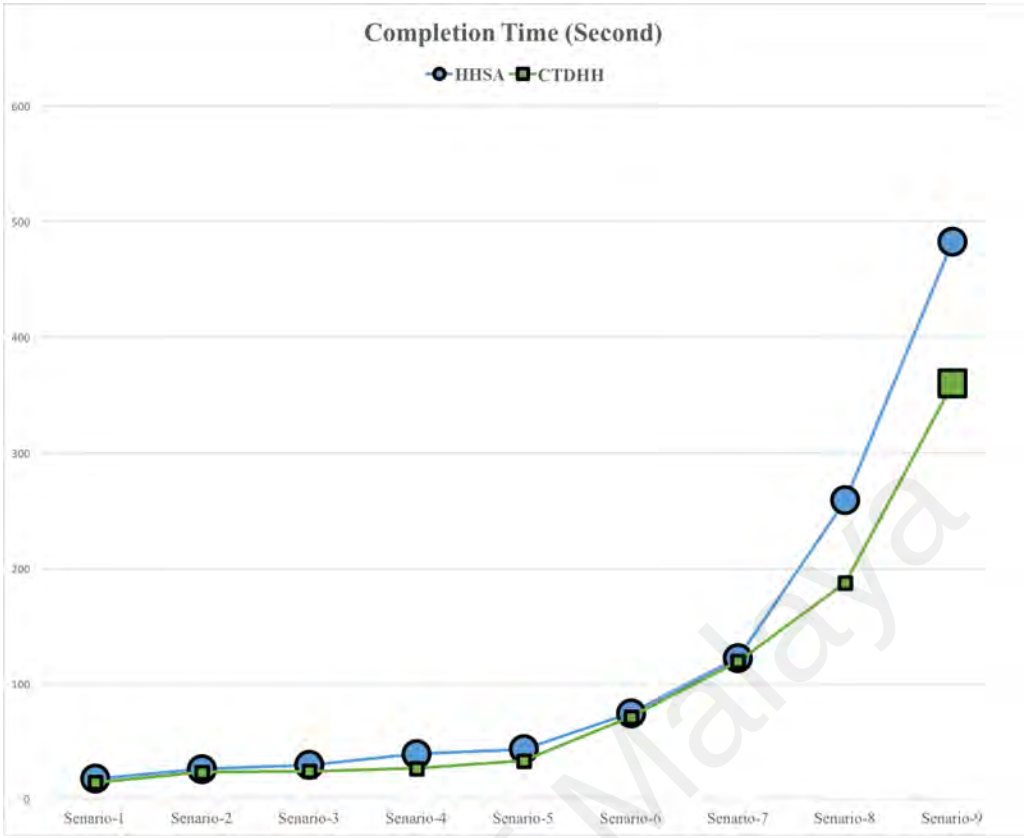

Figure 5.7: Average completion time of both approaches for Epigenomics SWFA

From Figure 5.7, it can be concluded that the proposed CTDHH approach has attained better performance than HHSA approach. Even if the number of tasks and VMs increases, it can be also observed that the average completion time of HHSA is significantly affected by the large numbers of workflow tasks (scenario seven, eight, and nine). Furthermore, for scenarios one to five, the differences between the average completion time values are marginally different for both approaches. This is mainly due to the fact that the employed LLH computes very a few number of optimal solutions for the smallest size scenarios (i.e., scenarios one to five). As a result, the hyper heuristic approaches have limited options to select the best LLH.

ii. Total computational cost results:

Table 5.6 represents the descriptive statistical analysis for total computational cost by comparing the proposed CTDHH approach and HHSA approach for Epigenomics SWFA.

Table 5.6: Total computational cost comparison CTDHH and HHSA for Epigenomics

Scenario-1		
	HHSA	**CTDHH**
AVERAGE	0.0205101	0.0167919
MIN	0.0003638	0.0069806
MAX	0.0308211	0.0308211
S.D.	0.0077409	0.0064994
Scenario-2		
	HHSA	**CTDHH**
AVAERAGE	0.0607612	0.0547907
MIN	0.0538072	0.0164304
MAX	0.0668576	0.0960991
S.D.	0.0061631	0.0169044
Scenario-3		
	HHSA	**CTDHH**
AVAERAGE	0.1381715	0.1128693
MIN	0.0891936	0.0618534
MAX	0.2163931	0.1852035
S.D.	0.037322	0.0339141
Scenario-4		
	HHSA	**CTDHH**
AVAERAGE	0.0458523	0.030969
MIN	0.0319172	0.0171393
MAX	0.0546123	0.0764225
S.D.	0.0092712	0.0146772
Scenario-5		
	HHSA	**CTDHH**
AVAERAGE	0.1010249	0.0777195
MIN	0.0530749	0.0281664
MAX	0.1578304	0.156915
S.D.	0.0325883	0.026113
Scenario-6		
	HHSA	**CTDHH**
AVERAGE	0.3493976	0.3331505
MIN	0.1128158	0.169374
MAX	0.5166281	0.4965736
S.D.	0.1271505	0.0757504
Scenario-7		
	HHSA	**CTDHH**
AVERAGE	0.143504	0.1400311
MIN	0.1286618	0.1285561
MAX	0.1565066	0.1565066
S.D.	0.010755	0.0107344
Scenario-8		
	HHSA	**CTDHH**
AVERAGE	0.6077121	0.439942
MIN	0.3285751	0.2989065
MAX	0.7188535	0.5104597
S.D.	0.1248352	0.101432
Scenario-9		
	HHSA	**CTDHH**
AVERAGE	2.2660723	1.6873836
MIN	0.8881335	1.4964621
MAX	3.3836109	1.9682493
S.D.	0.9325336	0.2078291

Table 5.6 shows that the total computational cost value depends on the completion time (makespan) value. This table shows that the values of total computational cost of larger datasets (scenarios seven, eight, and nine) are higher than those of the other smaller

datasets (scenarios one to six), this is due to the completion time required to execute the SWFA. Also, the total computational cost of the proposed CTDHH approach has the lowest average values for most of all scenarios compared to the HHSA approach.

Figure 5.8 illustrates the average total computational cost of CTDHH and HHSA for Epigenomics. This figure contains two charts, where each chart represents the relationship between the total computational cost ($/hour) and the considered scenarios.

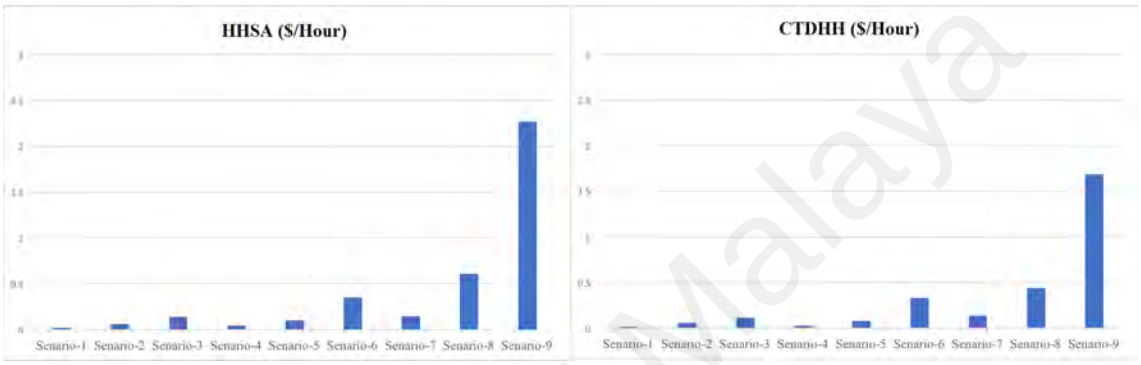

Figure 5.8: Average total computational cost of CTDHH and HHSA for Epigenomics

Figure 5.8 shows that the proposed CTDHH approach has more optimal performance for average total computational cost results for all of the considered scenarios. Furthermore, the average total computational cost results of the largest scenarios (scenarios seven, eight, and nine) are higher than those of the other scenarios (scenarios one to six), which reflects the similar results' trends as illustrated in Table 5.6.

Figure 5.9 illustrates the average total computational cost ($/hour) values of both approaches for the considered scenarios.

Figure 5.9: Average total computational of both approaches for Epigenomics SWFA

The results shown in Figure 5.9 pointed out that the HHSA approach does not show a good result comparing with the proposed CTDHH approach for Epigenomics SWFA. This is because of the random behavior of HHSA approach due to which HHSA cannot always determine the optimal results for larger scenarios (i.e., scenario seven, eight, and nine).

5.4.1.2 Inspiral Analysis (LIGO) - Gravitational Physics SWFAs

This section provides the result discussion of Inspiral SWFA. Table 5.7 shows the specification for utilized scenarios of Inspiral SWFA. Based on the datasets size, the first three scenarios are considered as the smallest datasets scenarios, while the last three scenarios are considered the largest datasets scenarios. Appendix B.2 describes the Inspiral SWFA datasets in detail.

This section has been divided into two parts. The first part presents the comparison

Table 5.7: Specification of scenarios for Inspiral SWFA in simulation environment

Scenario	No. of Tasks	No. of VMs
1	30	2
2	30	4
3	30	8
4	100	2
5	100	4
6	100	8
7	1000	2
8	1000	4
9	1000	8

between the proposed CTDHH approach and baseline approaches, while the second part presents the comparison between proposed CTDHH approach and HHSA approach.

A- Comparison between Proposed CTDHH Approach and Baseline Approaches

In this section, the descriptive statistical results for the completion time and total computational cost based on the comparison between proposed CTDHH approach and baseline approaches has been discussed.

i. Completion time results:

Table 5.8 represents the descriptive statistical analysis for completion time results for Inspiral SWFA for all considered nine scenarios.

Table 5.8: Completion time comparison between CTDHH and baselines for Inspiral

Scenario-1					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	3.45	3.449333333	3.45	3.45	2.979
MIN	3.45	3.43	3.45	3.45	1.74
MAX	3.45	3.45	3.45	3.45	4.048
S.D.	1.80672E-15	0.003651484	1.80672E-15	1.80672E-15	0.850991916
Scenario-2					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	7.41	9.285666667	7.965333333	7.285333333	5.9464
MIN	3.67	3.67	3.67	3.67	2.07
MAX	10.3	10.3	10.3	10.3	10.3
S.D.	2.984823682	1.800485918	2.912133633	2.666320897	2.271940104
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	19.35333333	18.62333333	17.73593333	17.967	14.79333333
MIN	7.7	7.7	7.53	4.11	7.7
MAX	26.7	26.7	26.7	26.7	22.9
S.D.	7.463369551	6.737014756	5.317247888	6.615523619	5.15972021
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**

AVERAGE	11.2	11.20466667	11.2	11.2	11.10733333
MIN	11.2	11.2	11.2	11.2	10.18
MAX	11.2	11.34	11.2	11.2	11.2
S.D.	5.42017E-15	0.025560386	5.42017E-15	5.42017E-15	0.210728348
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	23.91366667	25.31766667	24.29466667	25.5008	19.82233333
MIN	11.53	11.53	11.53	11.53	11.53
MAX	31.3	31.3	31.3	32.316	31.3
S.D.	8.230541727	7.945867147	8.700737873	7.080943082	6.685370381
Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	51.65133333	56.007	56.29566667	53.40926667	41.89833333
MIN	11.97	11.97	11.97	16.584	21.7
MAX	75.7	75.7	75.7	75.7	54.1
S.D.	20.24395589	19.82477658	18.8109179	20.6807493	13.13070161
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	110.2	110.4754	110.2	110.2	105.9406667
MIN	110.2	110.19	110.2	110.2	95.32
MAX	110.2	110.798	110.2	110.2	115.06
S.D.	4.33614E-14	0.300149457	4.33614E-14	4.33614E-14	7.346838839
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	269.1000667	222.788	215.98	257.0233333	205.3340667
MIN	73.79	110.2	110.2	200.3	110.2
MAX	301.616	300.4	300.4	300.4	300.4
S.D.	63.49477771	72.23347559	74.20156797	50.45109399	83.11686029
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	401.8333333	563.9966667	402.8333333	627.3933333	338.6054
MIN	110.2	200.3	110.2	500.6	158.68
MAX	700.8	700.8	700.8	700.8	702.832
S.D.	199.8865063	192.2468842	211.821005	98.12453007	162.5510125

Regarding the average completion time, it can be concluded from Table 5.8 that the proposed CTDHH approach performs much better than the other baseline approaches. This is due to the nature of Inspiral workforce tasks, which contains a less number of dependencies for smaller size of datasets. The Inspiral application is strongly affected by the smaller size of datasets, which gives limited options for the employed LLH algorithms to determine the SWFS solutions. This can be seen from the results of scenarios 1 to 7 for all the approaches. However, for larger size of datasets (scenario eight and nine), the differences between average completion time values are more than other scenarios.

Figure 5.10 illustrates the average completion time of the proposed CTDHH approach and baseline approaches for Inspiral SWFA. This figure contains two charts, where each chart represents the relationship between the average completion time (in second) and the considered scenarios.

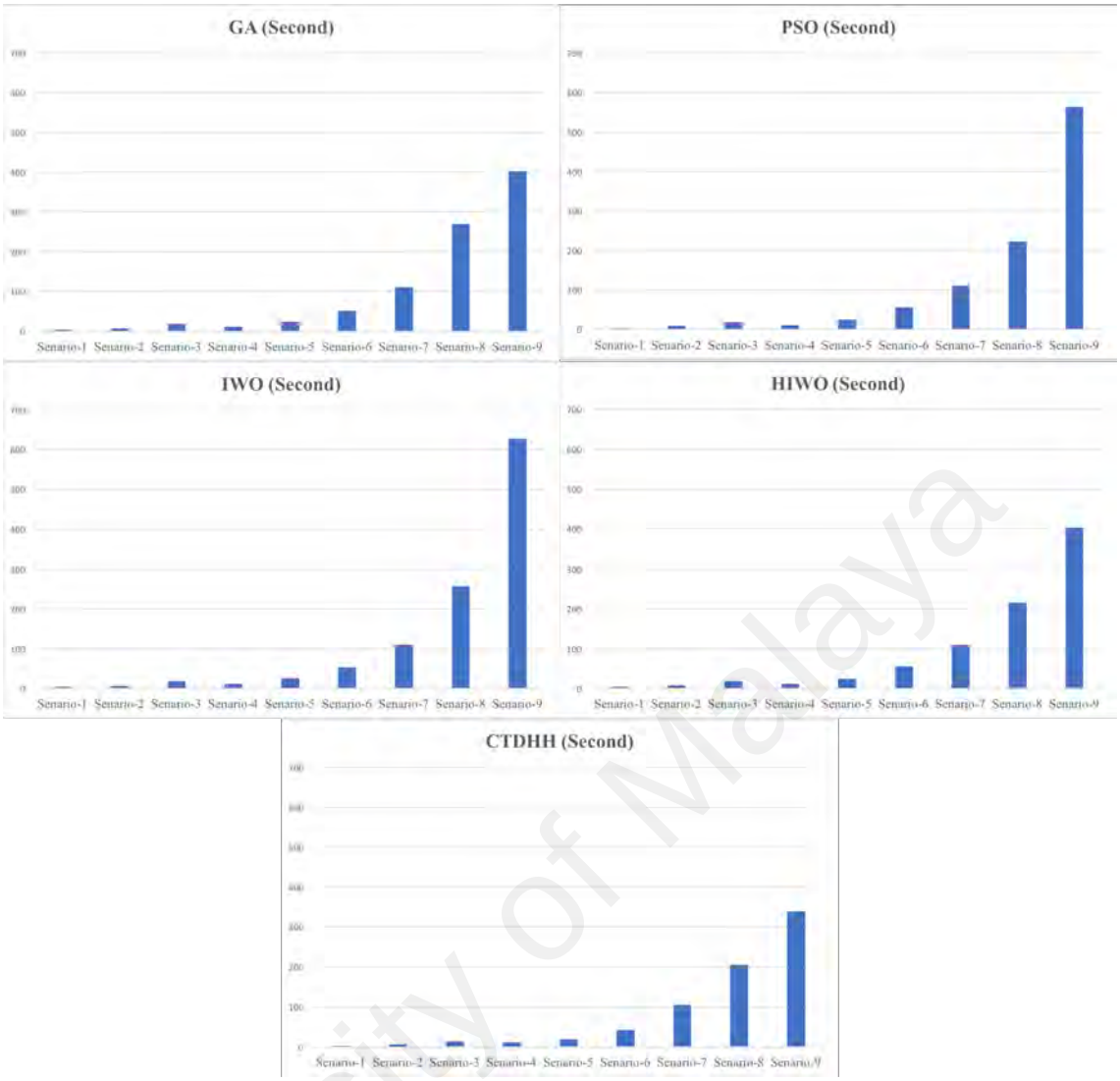

Figure 5.10: Average completion time of CTDHH and baselines for Inspiral SWFA

Figure 5.10 shows that the average completion time of the proposed CTDHH approach has a shorter completion time for all considered scenarios. At the same time, for scenarios one to seven (in Table 5.8), the baseline meta-heuristic algorithms attained very similar or close values to each other. The smaller size scenarios give limited options for the employed LLH algorithms to find SWFS solutions. In contrast, for scenarios that has larger size of tasks and large number of VMs, the approaches have larger search space to find the optimize solution.

Figure 5.11 shows the average completion time of all approaches for Inspiral SWFA of

all the considered scenarios.

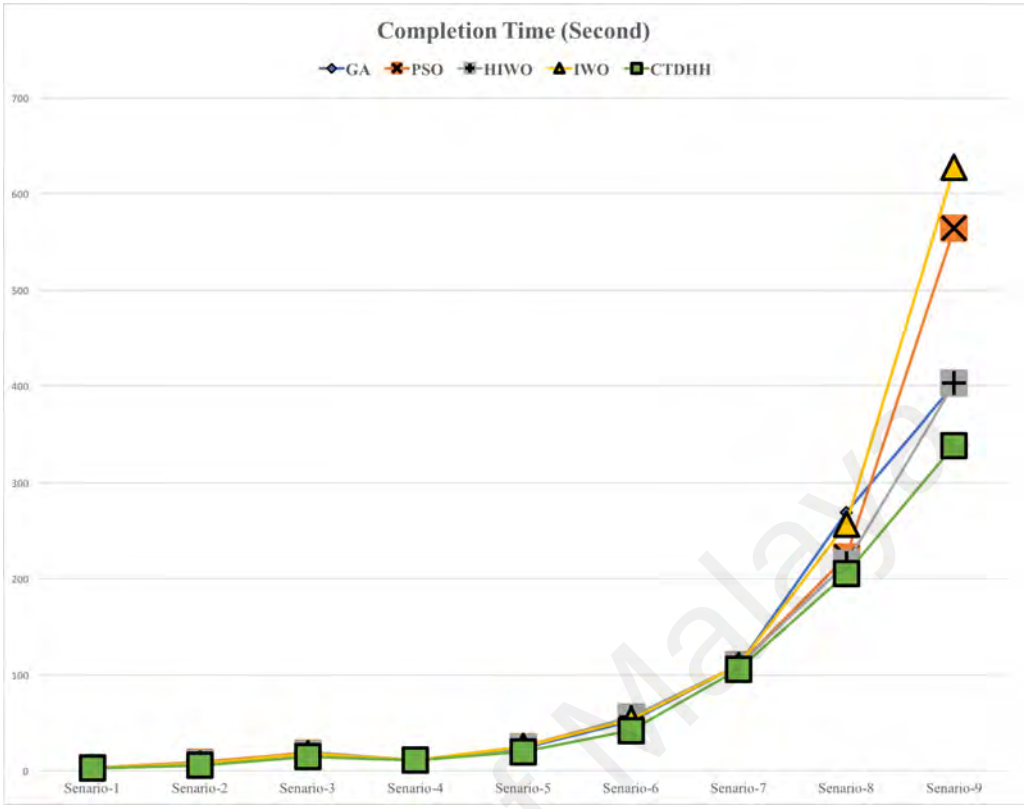

Figure 5.11: Average completion time of all approaches for Inspiral SWFA

From Figure 5.11, it can be noticed that the average completion time of the proposed CTDHH approach is lowest for scenarios eight and nine. In contrast, the average completion time values achieved by all approaches are very close to each other for scenarios one to seven.

ii. Total computational cost results

Table 5.9 represents the descriptive statistical analysis for total computational cost by comparing the proposed CTDHH approach and baseline approaches for Inspiral SWFA.

Table 5.9: Total computational cost comparison CTDHH and baselines for Inspiral

	Scenario-1				
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.00404892	0.004048138	0.00404892	0.00404892	0.003496154
MIN	0.00404892	0.004025448	0.00404892	0.00404892	0.002042064
MAX	0.00404892	0.00404892	0.00404892	0.00404892	0.004750733
S.D.	6.20289E-19	4.28538E-06	6.20289E-19	6.20289E-19	0.000998724
	Scenario-2				

	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	0.017392752	0.021795317	0.01869623	0.017100134	0.01395739
MIN	0.008614224	0.008614224	0.008614224	0.008614224	0.004858704
MAX	0.02417616	0.02417616	0.02417616	0.02417616	0.02417616
S.D.	0.007005978	0.004226101	0.00683536	0.006258388	0.005332698
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.090852288	0.087425376	0.083259565	0.084344285	0.069445824
MIN	0.03614688	0.03614688	0.035348832	0.019293984	0.03614688
MAX	0.12534048	0.12534048	0.12534048	0.12534048	0.10750176
S.D.	0.035036042	0.031626242	0.024961288	0.031055914	0.024221791
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.01314432	0.013149797	0.01314432	0.01314432	0.013035566
MIN	0.01314432	0.01314432	0.01314432	0.01314432	0.011947248
MAX	0.01314432	0.013308624	0.01314432	0.01314432	0.01314432
S.D.	9.92463E-18	2.99977E-05	9.92463E-18	9.92463E-18	0.000247311
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.056130158	0.059425627	0.057024442	0.059855478	0.046526981
MIN	0.027063216	0.027063216	0.027063216	0.027063216	0.027063216
MAX	0.07346736	0.07346736	0.07346736	0.075852115	0.07346736
S.D.	0.019318728	0.018650539	0.020422372	0.01662039	0.015691901
Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.242472019	0.262919261	0.264274378	0.250724461	0.196687536
MIN	0.056191968	0.056191968	0.056191968	0.07785193	0.10186848
MAX	0.35536608	0.35536608	0.35536608	0.35536608	0.25396704
S.D.	0.095033227	0.093065431	0.088305973	0.09708371	0.061640766
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.12933072	0.129653929	0.12933072	0.12933072	0.124331966
MIN	0.12933072	0.129318984	0.12933072	0.12933072	0.111867552
MAX	0.12933072	0.130032533	0.12933072	0.12933072	0.135034416
S.D.	5.95478E-17	0.000352255	5.95478E-17	5.95478E-17	0.00862225
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.631631676	0.522927994	0.506948256	0.603285168	0.481960121
MIN	0.173199888	0.25866144	0.25866144	0.47014416	0.25866144
MAX	0.707953075	0.70509888	0.70509888	0.70509888	0.70509888
S.D.	0.149034942	0.169546414	0.17416592	0.118418808	0.195091894
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	1.8863664	2.647625952	1.8910608	2.945235264	1.58954919
MIN	0.51732288	0.94028832	0.51732288	2.35001664	0.744907392
MAX	3.28983552	3.28983552	3.28983552	3.28983552	3.299374541
S.D.	0.938347215	0.902483773	0.994372526	0.460635794	0.763079473

From Table 5.9, it is obvious that the values of total computational cost of larger datasets (scenarios eight, and nine) are higher than those of the other smaller datasets (scenarios one to seven). This is due to the reason that for larger size datasets, the number of submitted workflow tasks and number of available VMs are larger, which gives more space for the baseline approaches to seach for the most optimal solution with minimal completion time and total computational cost.

Figure 5.12 illustrates the average total computational cost of the proposed CTDHH

approach and baseline approaches for Inspiral SWFA. Figure 5.12 shows five charts and each of these charts is representing the relationship between the total computational cost ($/hour) and the considered scenarios.

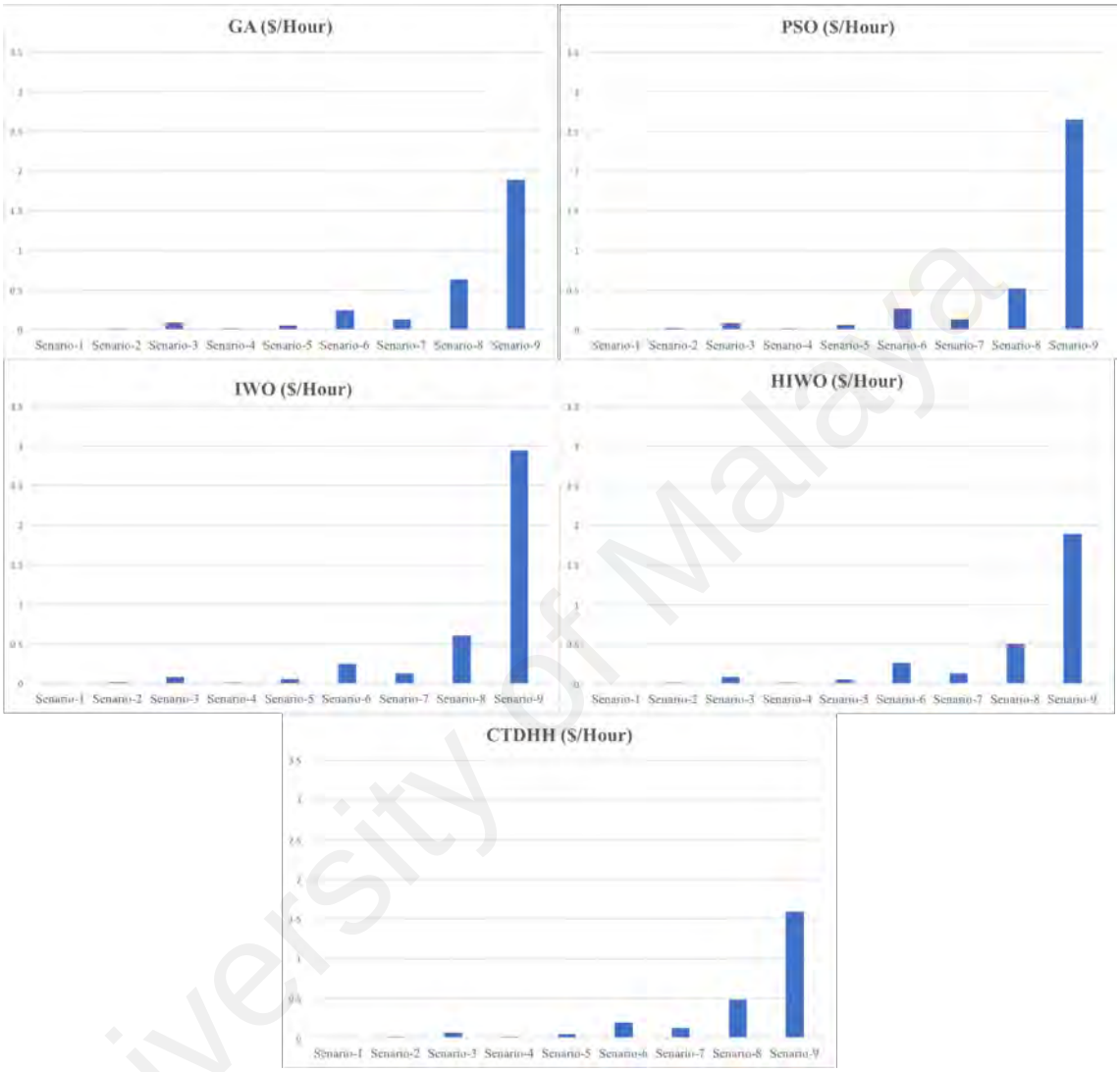

Figure 5.12: Average total computational cost of CTDHH and baselines for Inspiral

Figure 5.12 depicts that the proposed CTDHH approach has achieved optimal performance results for all considered scenarios. In addition, it can be also seen from this figure that the selection and approval operators of the proposed approach have successfully been utilized to dynamically select the most optimal LLH for each run and this ultimately affects the total computational cost. The underlying reason behind the close values of scenarios one to seven is that the dependencies between workflow tasks of this application are very

few, which gives very limited options for LLH to search for the optimal solution.

Figure 5.13 illustrates the average total computational cost ($/hour) values of all approaches for the considered scenarios.

Figure 5.13: Average total computational of all approaches for Inspiral SWFA

From Figure 5.13, it can be observed that the proposed approach gets the lowest total computational cost for all scenarios and especially for scenarios eight and nine.

B- Comparison between Proposed CTDHH Approach and HHSA Approach

In this section, the descriptive statistical results of completion time and computational cost based on the comparison between the proposed CTDHH approach and HHSA approach has been discussed.

i. Completion time results:

Table 5.10 illustrates the descriptive statistical analysis (average, minimum, maximum and S.D.) for completion time by comparing the proposed CTDHH approach and HHSA approach for Inspiral SWFA for all considered scenarios.

Table 5.10: Completion time comparison between CT-DHH and HHSA for Inspiral

Scenario-1		
	HHSA	**CTDHH**
AVERAGE	3.3188	2.979
MIN	1.85	1.74
MAX	4.048	4.048
S.D.	0.537644569	0.850991916
Scenario-2		
	HHSA	**CTDHH**
AVERAGE	8.024266667	5.9464
MIN	3.212	2.07
MAX	10.726	10.3
S.D.	2.08653871	2.271940104
Scenario-3		
	HHSA	**CTDHH**
AVERAGE	19.76426667	14.79333333
MIN	7.2	7.7
MAX	27.632	22.9
S.D.	5.271645866	5.15972021
Scenario-4		
	HHSA	**CTDHH**
AVERAGE	11.40626667	11.10733333
MIN	11.2	10.18
MAX	11.798	11.2
S.D.	0.280217026	0.210728348
Scenario-5		
	HHSA	**CTDHH**
AVERAGE	28.451	19.82233333
MIN	16.39	11.53
MAX	32.656	31.3
S.D.	5.40895803	6.685370381
Scenario-6		
	HHSA	**CTDHH**
AVERAGE	54.40866667	41.89833333
MIN	22.534	21.7
MAX	77.672	54.1
S.D.	18.65116373	13.13070161
Scenario-7		
	HHSA	**CTDHH**
AVERAGE	110.3186	105.9406667
MIN	110.2	95.32
MAX	110.798	115.06
S.D.	0.241265714	7.346838839
Scenario-8		
	HHSA	**CTDHH**
AVERAGE	251.3954	205.3340667
MIN	150.43	110.2
MAX	300.4	300.4
S.D.	60.46154092	83.11686029
Scenario-9		
	HHSA	**CTDHH**
AVERAGE	409.1258	338.6054
MIN	110.132	158.68
MAX	700.8	702.832
S.D.	203.2124665	162.5510125

From Table 5.10, it is obvious that there is small differences between proposed CTDHH approach and HHSA approach in terms of computed average completion time value for all considered scenarios. Moreover, there are some scenarios for which the proposed approach

achieved a slightly higher S.D. than that of the HHSA approach, this might be due to the HLH strategy, which ultimately affects the selection of the employed LLH after each run.

Figure 5.14 illustrates the average completion time of CTDHH and HHSA for Inspiral SWFA. This figure has two charts and each of these charts is representing the relationship between the completion time and the considered scenarios.

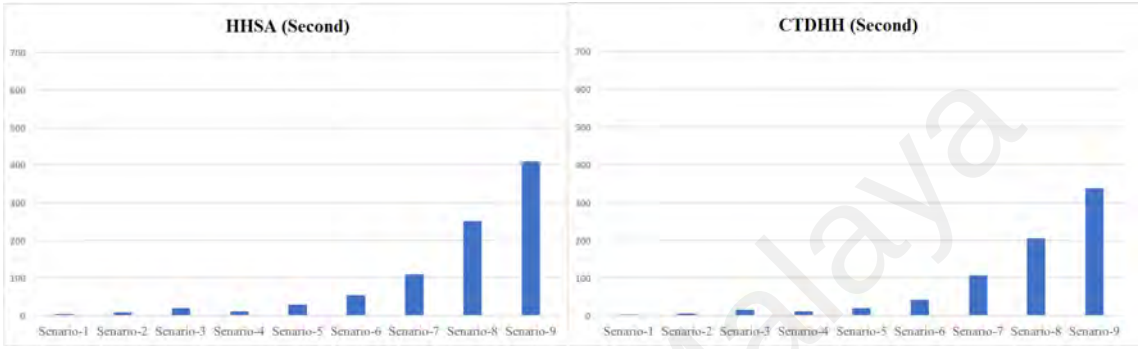

Figure 5.14: Average completion time of CTDHH and HHSA for Inspiral SWFA

From Figure 5.14, it can be seen that average completion time of the proposed CTDHH approach is slightly better for all considered scenarios comparing with completion time of HHSA approach.

Figure 5.15 shows the average completion time of both approaches for Inspiral SWFA.

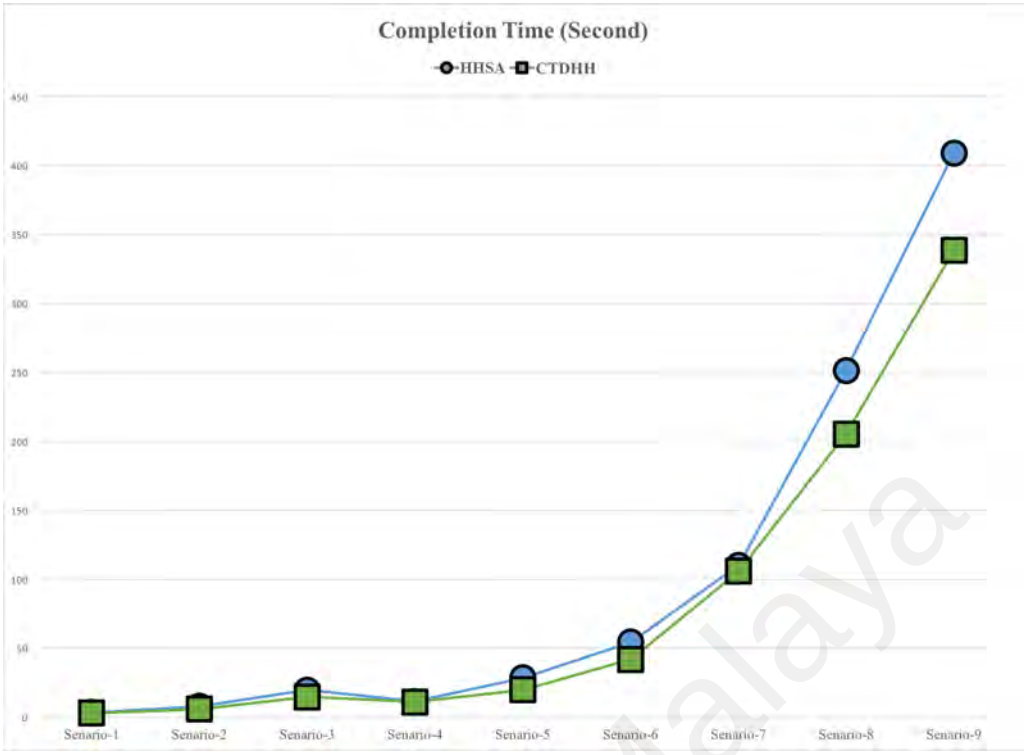

Figure 5.15: Average completion time of both approaches for Inspiral SWFA

From Figure 5.15, it can be concluded that the average completion time of the proposed CTDHH approach and HHSA are very close to each other especially for scenarios one to seven.

ii. Total computational cost results

Table 5.11 represents the descriptive statistical analysis for total computational cost when comparing the proposed CTDHH approach with HHSA approach for Inspiral SWFA.

Table 5.11: Total computational cost comparison CT-DHH and HHSA for Inspiral

Senario-1		
	HHSA	**CTDHH**
AVERAGE	0.003894944	0.003496154
MIN	0.00217116	0.002042064
MAX	0.004750733	0.004750733
S.D.	0.00063098	0.000998724
Senario-2		
	HHSA	**CTDHH**
AVERAGE	0.018834559	0.01395739
MIN	0.007539206	0.004858704
MAX	0.025176067	0.02417616
S.D.	0.004897524	0.005332698
Senario-3		
	HHSA	**CTDHH**

AVERAGE	0.092781373	0.069445824
MIN	0.03379968	0.03614688
MAX	0.129715661	0.10750176
S.D.	0.024747214	0.024221791
Senario-4		
	HHSA	**CTDHH**
AVERAGE	0.013386395	0.013035566
MIN	0.01314432	0.011947248
MAX	0.013846133	0.01314432
S.D.	0.000328863	0.000247311
Senario-5		
	HHSA	**CTDHH**
AVERAGE	0.066780187	0.046526981
MIN	0.038470608	0.027063216
MAX	0.076650163	0.07346736
S.D.	0.012695906	0.015691901
Senario-6		
	HHSA	**CTDHH**
AVERAGE	0.255416045	0.196687536
MIN	0.10578361	0.10186848
MAX	0.364623437	0.25396704
S.D.	0.087556023	0.061640766
Senario-7		
	HHSA	**CTDHH**
AVERAGE	0.129469909	0.124331966
MIN	0.12933072	0.111867552
MAX	0.130032533	0.135034416
S.D.	0.000283149	0.00862225
Senario-8		
	HHSA	**CTDHH**
AVERAGE	0.590075283	0.481960121
MIN	0.353089296	0.25866144
MAX	0.70509888	0.70509888
S.D.	0.141915329	0.195091894
Senario-9		
	HHSA	**CTDHH**
AVERAGE	1.920600156	1.58954919
MIN	0.517003661	0.744907392
MAX	3.28983552	3.299374541
S.D.	0.953960603	0.763079473

From Table 5.11, it can be seen that the total computational cost of the proposed CTDHH approach has the lowest average value for most of the scenarios compared to that of the HHSA approach.

Figure 5.16 illustrates the average total computational cost of CTDHH and HHSA approach. Figure 5.16 contains two charts and each of these charts is representing the relationship between the total computational cost ($/hour) and the considered scenarios.

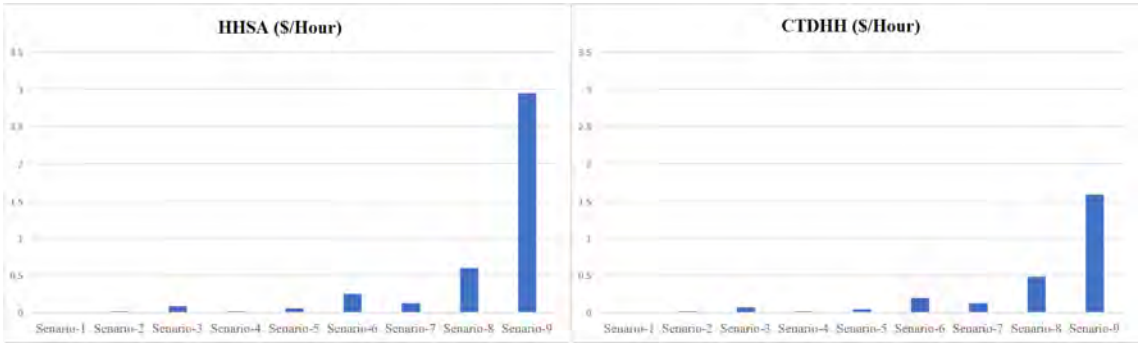

Figure 5.16: Average total computational cost of CTDHH and HHSA for Inspiral

Figure 5.16 shows that the total computational cost of both approaches are very close to each other for scenarios one to eight. This is due to the reason that the number of workflow tasks in this scenario is 1000 and the number of available VMs are eight, which gives the selection and approaval operators of the HLH more options to select which of the employed LLH algorithms is more appropriate for each run. Generally speaking, a higher number of tasks and available VMs provide the facility to selection and approval operators of the HLH to decide about the best LLH for each execution of the approach.

Figure 5.17 illustrates the average total computational cost ($/hour) values of both approaches for the considered scenarios.

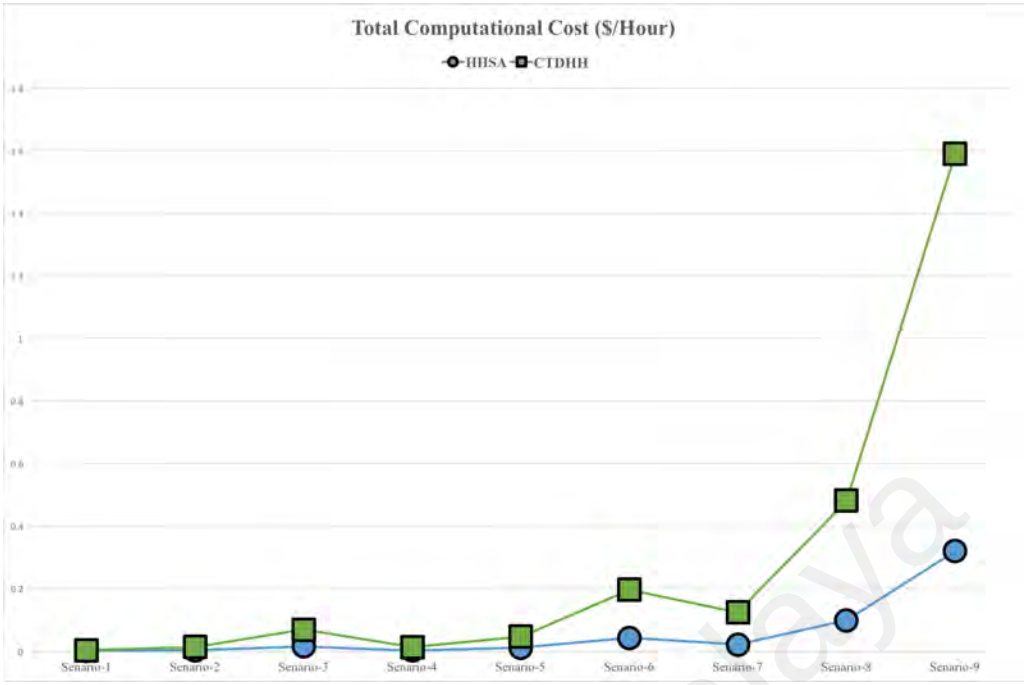

Figure 5.17: Average total computational of both approaches for Inspiral SWFA

From Figure 5.17, it is evident that there is a significant differences between the performance of SWFS approaches for scenario six to nine, which is ultimately affected by the size of dataset of the submitted SWFA as well as the number of available VMs. This is because of the random nature of HHSA approach due to which it cannot always get the optimal result by selecting the most optimal solution of the employed LLH approaches.

5.4.1.3 Montage SWFAs

This section provides the discussion about the achieved results of Montage SWFA. Table 5.12 shows the specification of each of the utilized scenarios of Montage SWFA. Based on the number of tasks, the first three scenarios are considered as the smallest datasets scenarios, while the last three scenarios are considered as the largest datasets scenarios. Appendix B.3 describes the Montage SWFA datasets in detail.

In the following sections, the statistical results of completion time and total computational cost for each of the considered nine scenarios for Montage SWFA have been provided.

Table 5.12: Specification of scenarios for Montage SWFA in simulation environment

Scenario	No. of Tasks	No. of VMs
1	25	2
2	25	4
3	25	8
4	100	2
5	100	4
6	100	8
7	1000	2
8	1000	4
9	1000	8

A- Comparison between Proposed CTDHH Approach and Baseline Approaches

This section provides a discussion about the completion time and total computational cost statistical results based on the comparison between the proposed CTDHH approach and baseline approaches for each of the nine considered scenarios.

i. Completion time results:

Table 5.13 represents the descriptive statistical analysis for completion time results for Montage SWFA.

Table 5.13: Completion time comparison between CT-DHH and baselines for Montage

Scenario-1					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	2.95	2.95	2.95	2.95	2.309333333
MIN	2.95	2.95	2.95	2.95	1.774
MAX	2.95	2.95	2.95	2.95	2.95
S.D.	1.35504E-15	1.35504E-15	1.35504E-15	1.35504E-15	0.423633001
Scenario-2					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	5.865666667	6.183	7.179333333	6.986	5.214266667
MIN	3.17	3.17	3.17	3.17	3.052
MAX	8.8	8.8	8.8	8.8	8.8
S.D.	2.392852407	2.164188388	2.065152467	2.039098178	1.196539001
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	15.067	15.184	16.16	13.328	11.8564
MIN	3.61	3.61	6.7	3.61	4.69
MAX	23.2	23.2	23.2	23.2	23.2
S.D.	6.685386611	6.836611683	5.98743512	7.467217734	4.407987496
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	11.09	11.00666667	11.09	11.09	9.8944
MIN	11.09	9.05	11.09	11.09	8.77
MAX	11.09	11.09	11.09	11.09	11.134
S.D.	3.61345E-15	0.374122729	3.61345E-15	3.61345E-15	0.970393934
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	20.53066667	24.162	27.38306667	21.224	20.157
MIN	11.31	11.31	9.006	11.31	11.31

MAX	31.3	31.3	49.728	31.3	31.3
S.D.	8.672439474	7.998597463	7.844059993	9.094120723	8.009407464
Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	53.02	55.20333333	52.66	47.67666667	44.86113333
MIN	21.7	11.75	21.7	11.75	23.186
MAX	75.7	75.7	75.7	75.7	75
S.D.	17.56395997	20.19728956	20.00154477	21.79844084	13.02581149
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	108.59	108.59	108.59	108.59	96.46046667
MIN	108.59	108.59	108.59	108.59	91.26
MAX	108.59	108.59	108.59	108.59	108.49
S.D.	4.33614E-14	4.33614E-14	4.33614E-14	4.33614E-14	7.863931871
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	254.7544	239.8543333	219.5723333	217.3336667	207.0166667
MIN	97.986	108.21	108.81	108.81	108.81
MAX	301.3	301.3	301.3	301.3	301.3
S.D.	66.97655323	69.10706529	76.53401672	90.28435674	81.90933437
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	612.8208	495.8310667	490.9383333	387.6133333	451.4533333
MIN	201.7	101.282	109.25	109.25	109.25
MAX	705.7	705.7	705.7	604.9	705.7
S.D.	148.5438475	198.9182474	204.0466646	191.1746723	214.3961741

From Table 5.13, it can be observed that there are more differences between the average completion time of all approaches. This is due to the strong dependencies between the workflow tasks of Montage SWFA, which ultimately gives wider space for employed LLH algorithms of the proposed approach to search for the optimal solution. Furthermore, it can be also observed that the average completion time of the proposed approach has achieved the most optimal results for all scenarios except scenario nine. Moreover, the proposed approach has achived acceptable S.D. results for all scenarios. This is due to large size of datasets and large of available VMs (i.e., 8), which give more space for the proposed approach to find the most optimal solution.

Figure 5.18 illustrates the average completion time of the proposed CTDHH approach and baseline approaches for Montage SWFA. This figure has five charts and each of these charts is representing the relationship between the completion time and the considered scenarios for each of the approaches.

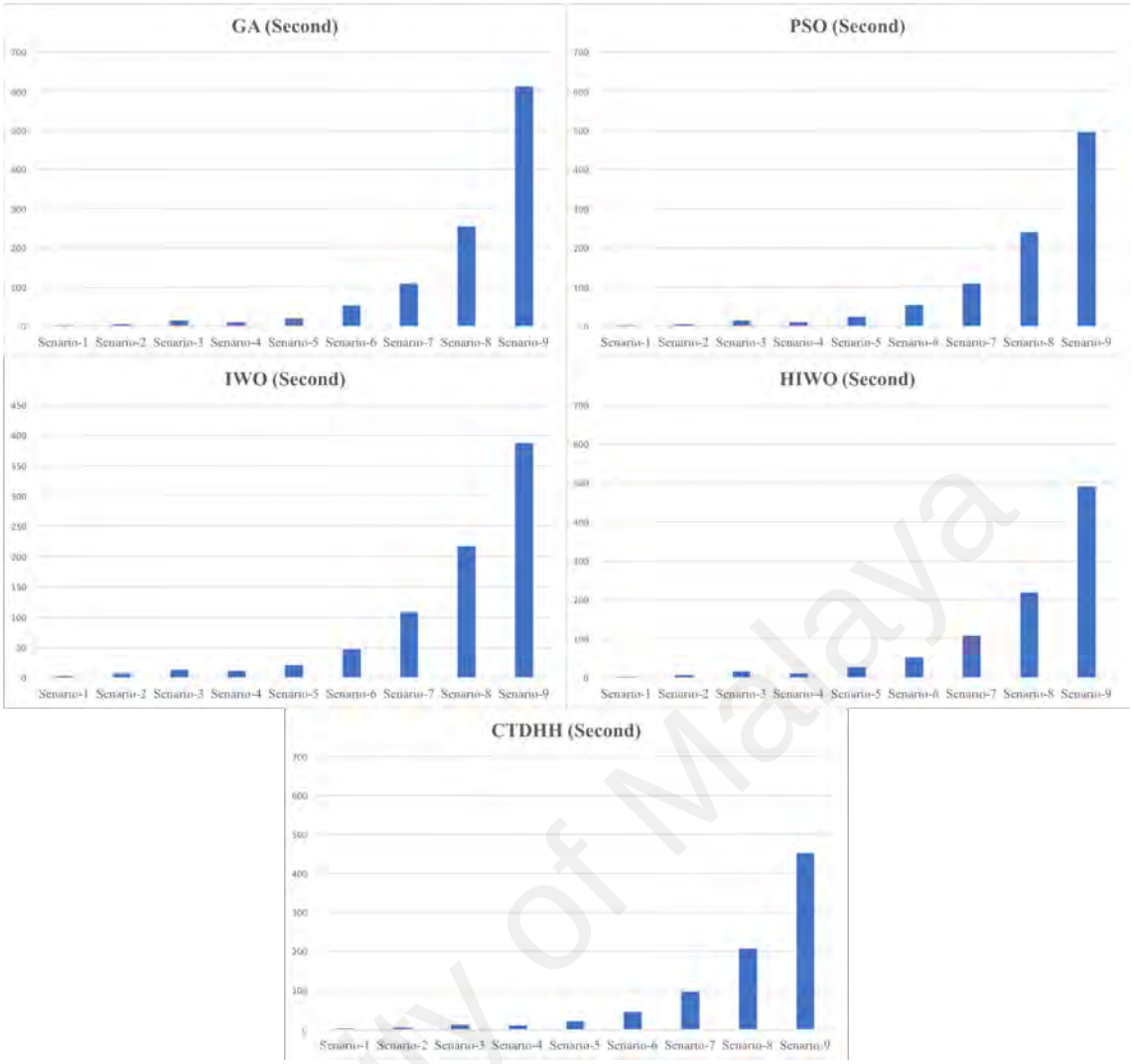

Figure 5.18: Average completion time of CTDHH and baselines for Montage SWFA

Figure 5.18 depicts that the average completion time (in second) increases as the number of workflow tasks and number of available VMs increase for all approaches. The average completion time of the proposed CTDHH approach has the lowest results for most of the considered scenarios. However, the baseline approaches always have very similar or close values to each other.

Figure 5.19 shows the average completion time of all approaches for Montage SWFA of all considered scenarios.

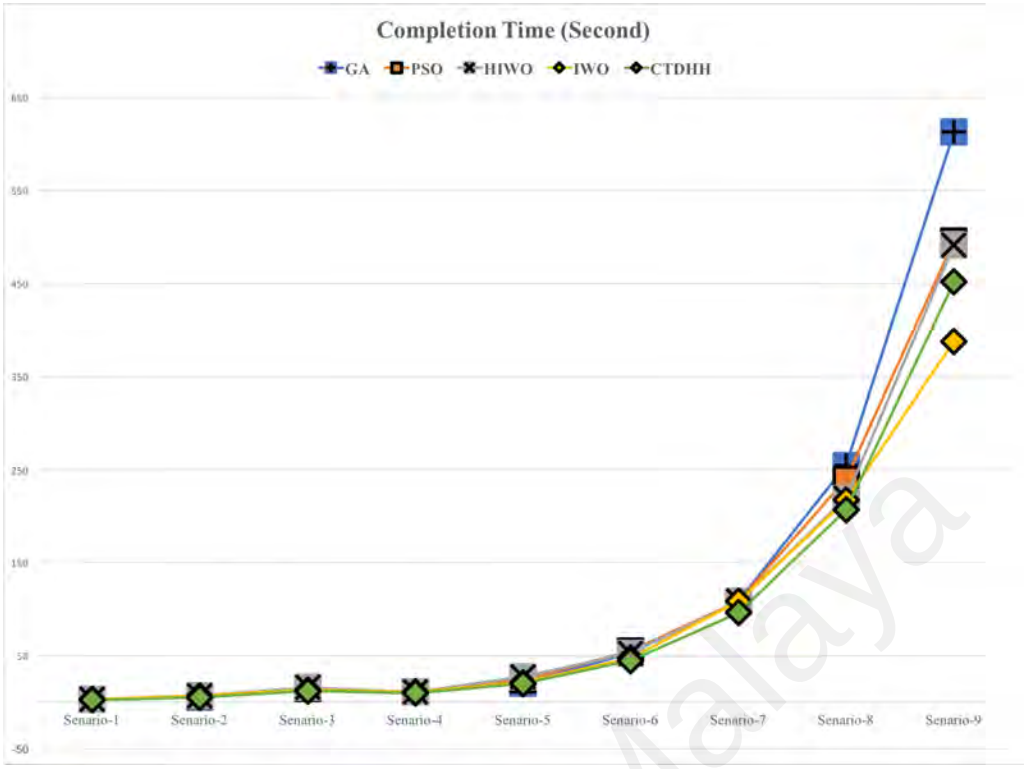

Figure 5.19: Average completion time of all approaches for Montage SWFA

From Figure 5.19, it can be concluded that as the size of dataset increases (with several available VMs), the proposed CTDHH approach and IWO performs better than other approaches. Furthermore, for scenarios one to five, the differences between the average completion time values are relatively small for all approaches.

ii. Total computational cost results:

Table 5.14 represents the descriptive statistical analysis for total computational cost by comparing the proposed CTDHH approach and baseline approaches for Montage SWFA for all considered scenarios.

Table 5.14: Total computational cost comparison CT-DHH and baselines for Montage

Scenario-1					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.00346212	0.00346212	0.00346212	0.00346212	0.002710234
MIN	0.00346212	0.00346212	0.00346212	0.00346212	0.002081966
MAX	0.00346212	0.00346212	0.00346212	0.00346212	0.00346212
S.D.	2.48116E-18	2.48116E-18	2.48116E-18	2.48116E-18	0.000497176
Scenario-2					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**

AVERAGE	0.015049464	0.014512738	0.016851331	0.016397539	0.012238927
MIN	0.007440624	0.007440624	0.007440624	0.007440624	0.007163654
MAX	0.02065536	0.02065536	0.02065536	0.02065536	0.02065536
S.D.	0.004643751	0.005079783	0.004847326	0.004786171	0.002808516
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.078959808	0.07127977	0.075861504	0.0727632	0.055658684
MIN	0.03145248	0.016946784	0.03145248	0.03145248	0.022016736
MAX	0.10891008	0.10891008	0.10891008	0.10891008	0.10891008
S.D.	0.029317918	0.03209379	0.028107415	0.031071944	0.020692857
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.013015224	0.012917424	0.013015224	0.013015224	0.011612068
MIN	0.013015224	0.01062108	0.013015224	0.013015224	0.010292472
MAX	0.013015224	0.013015224	0.013015224	0.013015224	0.013066862
S.D.	7.44347E-18	0.00043907	7.44347E-18	7.44347E-18	0.001138854
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.048189581	0.056713046	0.064273534	0.049816973	0.043301928
MIN	0.026546832	0.026546832	0.021138883	0.026546832	0.026546832
MAX	0.07346736	0.07346736	0.116721562	0.07346736	0.054947952
S.D.	0.02035595	0.018774308	0.018411578	0.02134572	0.008008016
Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.248897088	0.259146528	0.247207104	0.2370672	0.191367842
MIN	0.10186848	0.0551592	0.10186848	0.10186848	0.108844358
MAX	0.35536608	0.35536608	0.35536608	0.35536608	0.28307232
S.D.	0.082452254	0.094814156	0.093895252	0.08559927	0.044669991
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.127441224	0.127441224	0.127441224	0.127441224	0.113206004
MIN	0.127441224	0.127441224	0.127441224	0.127441224	0.107102736
MAX	0.127441224	0.127441224	0.127441224	0.127441224	0.127323864
S.D.	9.92463E-17	9.92463E-17	9.92463E-17	9.92463E-17	0.00922911
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.597959528	0.562986091	0.515380181	0.510125582	0.430317653
MIN	0.229992739	0.253990512	0.255398832	0.255398832	0.255398832
MAX	0.70721136	0.70721136	0.70721136	0.70721136	0.49455504
S.D.	0.157207366	0.162208104	0.179640644	0.211915442	0.098680411
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	2.876825964	2.327629359	2.304660912	2.256034752	1.819612032
MIN	0.94686048	0.475458221	0.5128632	0.94686048	0.5128632
MAX	3.31283808	3.31283808	3.31283808	2.83964256	2.36644704
S.D.	0.697324238	0.933801821	0.957876662	0.72121401	0.704677437

It is evident from Table 5.14 that the total computational cost of the proposed CTDHH approach has the lowest average value for most of the scenarios comparing with other baseline approaches. This is due to the reason that total computational cost is related to the time spend by the approach to compete the submitted SWFA.

Figure 5.20 illustrates the average total computational cost of CTDHH and baselines for Montage SWFA. This figure has five charts and each of these charts is representing the relationship between the total computational cost ($/hour) and the considered scenarios.

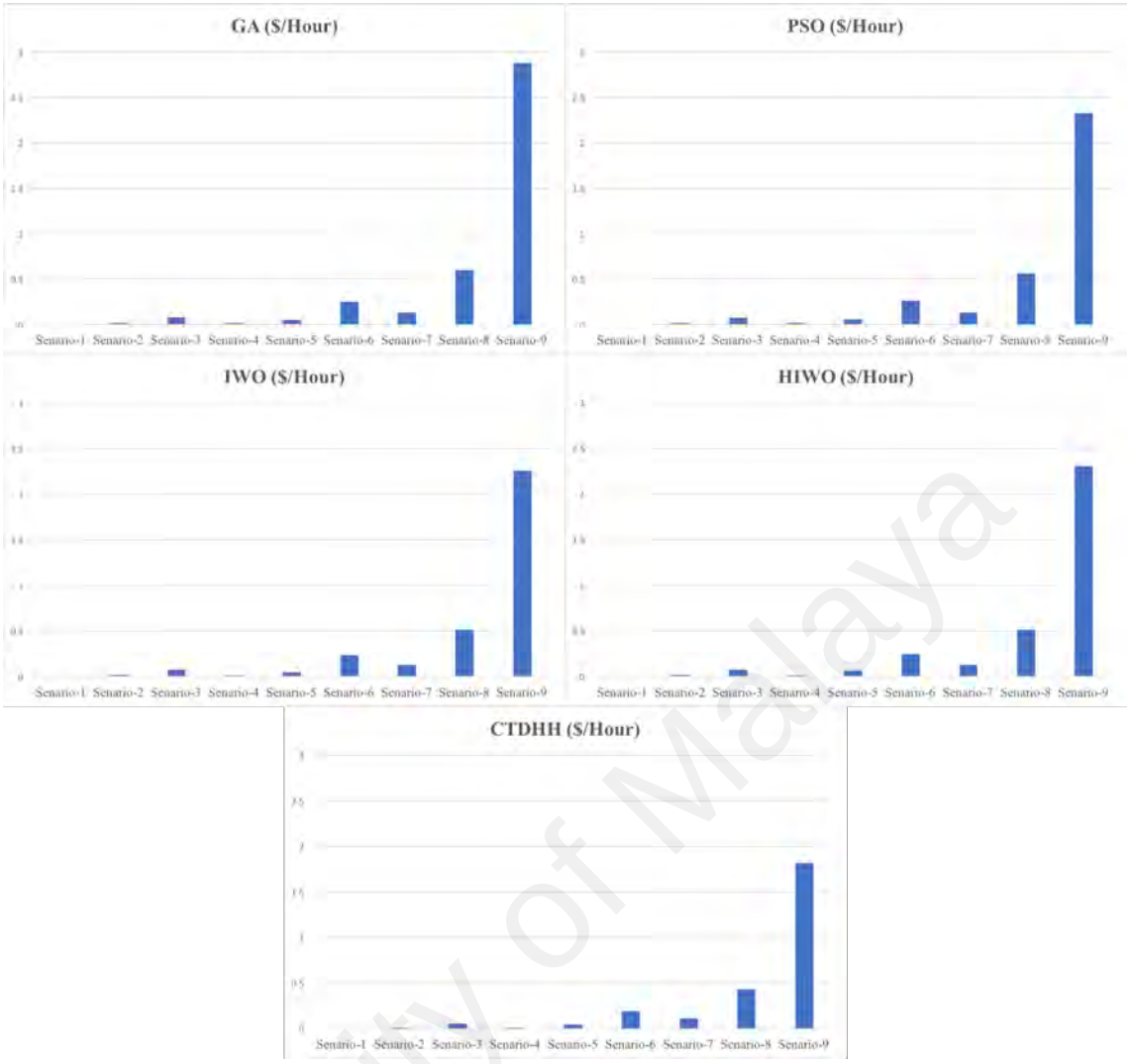

Figure 5.20: Average total computational cost of CTDHH and baselines for Montage

Figure 5.20 shows that the total computational cost heavily depends on the completion time, number of available VMs as well as the number of workflow tasks for each of the considered scenarios. It can be also observed from the charts (Figure 5.20) that the average total computational cost of this application is much less than the Epigenomics and Inspiral SWFAs even using the same number of workflow tasks. This mainly occurs due to the nature of dataset that has been executed by Montage SWFA.

Figure 5.21 illustrates the average total computational cost ($/hour) values of all approaches for the considered scenarios.

Figure 5.21: Average total computational of all approaches for Montage SWFA

From Figure 5.21, it can be seen that there is a significant differences between the performance of SWFS approaches for scenarios eight and nine. This has confirmed the argument that the cost results are dependent on the average completion values, which are ultimately affected by the size of dataset of the submitted SWFA as well as the number of available VMs.

B- Comparison between Proposed CTDHH Approach and HHSA Approach

In this section, the completion time and total computational cost statistical results have been discussed, which are based on the comparison between the proposed CTDHH approach and HHSA approach for each of the nine considered scenarios.

i. Completion time results:

Table 5.15 illustrates the descriptive statistical analysis for completion time by comparing the proposed CTDHH approach and HHSA approach for Montage SWFA for all considered scenarios.

Table 5.15: Completion time comparison between CTDHH and HHSA for Montage

Scenario-1		
	HHSA	**CTDHH**
AVERAGE	2.7508	2.309333333
MIN	2.004	1.774
MAX	2.994	2.95
S.D.	0.282925407	0.423633001
Scenario-2		
	HHSA	**CTDHH**
AVERAGE	5.972133333	5.214266667
MIN	4.452	3.052
MAX	7.9	8.8
S.D.	0.973905318	1.196539001
Scenario-3		
	HHSA	**CTDHH**
AVERAGE	18.3238	11.8564
MIN	8.584	4.69
MAX	63.68	23.2
S.D.	10.42268274	4.407987496
Scenario-4		
	HHSA	**CTDHH**
AVERAGE	10.73933333	9.8944
MIN	8.89	8.77
MAX	11.09	11.134
S.D.	0.723308173	0.970393934
Scenario-5		
	HHSA	**CTDHH**
AVERAGE	23.46173333	18.44833333
MIN	13.824	11.31
MAX	31	23.41
S.D.	4.771669779	3.411731367
Scenario-6		
	HHSA	**CTDHH**
AVERAGE	47.20933333	40.76513333
MIN	30.348	23.186
MAX	72.2	60.3
S.D.	16.02239557	9.515591035
Scenario-7		
	HHSA	**CTDHH**
AVERAGE	101.4836	96.46046667
MIN	91.26	91.26
MAX	108.59	108.49
S.D.	8.358356799	7.863931871
Scenario-8		
	HHSA	**CTDHH**
AVERAGE	242.0149333	183.3323333
MIN	91.798	108.81
MAX	301.3	210.7
S.D.	70.41136552	42.04175647
Scenario-9		
	HHSA	**CTDHH**
AVERAGE	446.2908	387.6133333
MIN	199.562	109.25
MAX	705.7	504.1
S.D.	166.490772	150.1102244

From Table 5.15, it can be observed that the average completion time of both approaches (i.e., CTDHH and HHSA) are very close to each other, which shows that both hyper-heuristic approaches are achieving optimised results. However, the average completion

time and S.D. values of the proposed CTDHH are more optimal due to the dynamic strategy of the HLH by continuously learning from the computed time score after each run based on the selection and approval operators of HLH strategy.

Figure 5.22 illustrates the average completion time of CTDHH and HHSA approach for Montage SWFA. This figure contains two charts and each of these charts is representing the relationship between the completion time and the considered scenarios for each of the approaches.

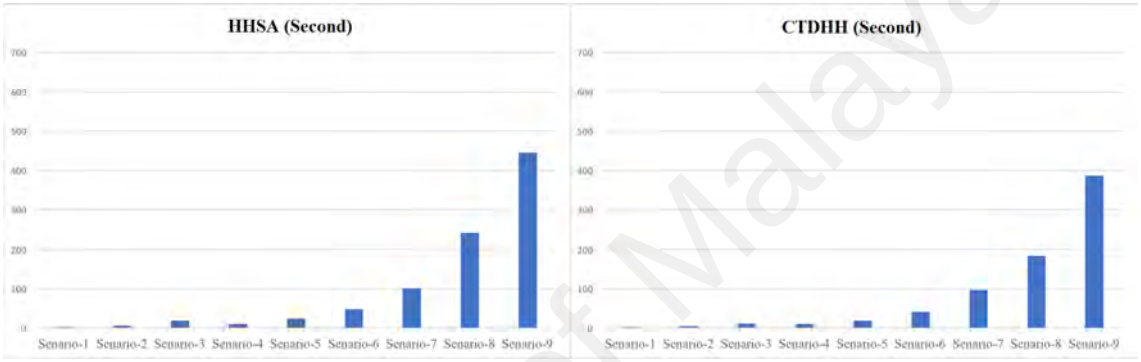

Figure 5.22: Average completion time of CTDHH and HHSA for Montage SWFA

Regarding completion time of Montage SWFA, it can be concluded from Figure 5.22 that the proposed CTDHH approach achieved much lower values than the HHSA approach for all scenarios one to seven, while there is a major improvement in the performance of the proposed CTDHH approach for scenarios eight and nine.

Figure 5.23 shows the average completion time of both approaches for Montage SWFA of all considered scenarios.

Figure 5.23: Average completion time of both approaches for Montage SWFA

The results shown in Figure 5.23 have confirmed the observation depicted by Figure 5.22 that for scenarios eight and nine, the proposed CTDHH approach achieves clear improvement when comparing with the HHSA approach; while for other scenarios, both approaches achieve almost equal optimisation results.

ii. Total computational cost results:

Table 5.16 represents the descriptive statistical analysis for total computational cost by comparing the proposed CTDHH approach and HHSA approach for Montage SWFA.

Table 5.16: Total computational cost comparison CT-DHH and HHSA for Montage

Scenario-1		
	HHSA	**CTDHH**
AVERAGE	0.003228339	0.002710234
MIN	0.002351894	0.002081966
MAX	0.003513758	0.00346212
S.D.	0.000332041	0.000497176
Scenario-2		
	HHSA	**CTDHH**
AVERAGE	0.014017791	0.012238927
MIN	0.010449734	0.007163654
MAX	0.01854288	0.02065536

S.D.	0.002285951	0.002808516
	Scenario-3	
	HHSA	**CTDHH**
AVERAGE	0.086019247	0.055658684
MIN	0.04029673	0.022016736
MAX	0.298939392	0.10891008
S.D.	0.048928242	0.020692857
	Scenario-4	
	HHSA	**CTDHH**
AVERAGE	0.012603682	0.011612068
MIN	0.010433304	0.010292472
MAX	0.013015224	0.013066862
S.D.	0.000848874	0.001138854
	Scenario-5	
	HHSA	**CTDHH**
AVERAGE	0.05506938	0.043301928
MIN	0.032447693	0.026546832
MAX	0.0727632	0.054947952
S.D.	0.011200063	0.008008016
	Scenario-6	
	HHSA	**CTDHH**
AVERAGE	0.221619494	0.191367842
MIN	0.142465651	0.108844358
MAX	0.33893568	0.28307232
S.D.	0.075215534	0.044669991
	Scenario-7	
	HHSA	**CTDHH**
AVERAGE	0.119101153	0.113206004
MIN	0.107102736	0.107102736
MAX	0.127441224	0.127323864
S.D.	0.009809368	0.00922911
	Scenario-8	
	HHSA	**CTDHH**
AVERAGE	0.568057452	0.430317653
MIN	0.215468266	0.255398832
MAX	0.70721136	0.49455504
S.D.	0.165269557	0.098680411
	Scenario-9	
	HHSA	**CTDHH**
AVERAGE	2.095067532	1.819612032
MIN	0.936823853	0.5128632
MAX	3.31283808	2.36644704
S.D.	0.78157428	0.704677437

In terms of computational cost, it can be concluded from Table 5.16 that the proposed CTDHH approach has the lowest average value for all scenarios compared to other baseline approaches. This is because the total computational cost of SWFS is strongly depends on the completion time (makespan) required by the proposed approach to complete execution of the submitted SWFAs (based on the problem statement of this research).

Figure 5.24 illustrates the average total computational cost of CTDHH and HHSA approach. This figure has two charts and each of these charts is representing the relationship between the total computational cost ($/hour) and the considered scenarios for each of the approaches.

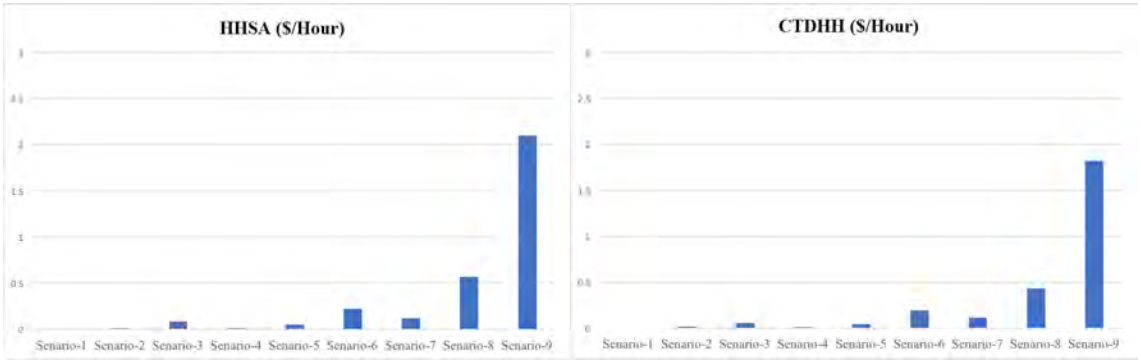

Figure 5.24: Average total computational cost of CTDHH and HHSA for Montage

Figure 5.24 shows that the total computational cost heavily depends on the completion time. The proposed CTDHH approach shows similar results for scenarios eight and nine. In these two scenarios, the proposed approach managed to achieve the optimal result due to the high number of tasks (i.e., 1000 tasks) and the available VMs (8 VMs). In this case, the chances of finding the optimal solution of the proposed approach is higher compared to other scenarios (i.e., scenarios one to seven) where the number of tasks and VMs are lower.

Figure 5.25 illustrates the average total computational cost ($/hour) values of both approaches for the considered scenarios.

Figure 5.25: Average total computational of both approaches for Montage SWFA

Similar to the previous figure (Figure 5.24), Figure 5.25 shows that there is a major differences between the total computational cost value of the proposed CTDHH and HHSA approach.

5.4.1.4 SIPHT-search for sRNAs SWFAs:

This section provides the result discussion of SIPHT SWFA. Table 5.17 shows the specification of each of the utilized scenarios of SIPHT SWFA. Based on the workflow tasks (datasets size), the first three scenarios are considered as the smallest datasets scenarios, while the last three scenarios are considered as the largest datasets scenarios. Appendix B.4 describes the SIPHT SWFA datasets in detail.

This section has been divided into two parts. The first part presents the comparison between proposed CTDHH approach and baseline approaches, while the second part presents the comparison between proposed CTDHH approach and HHSA approach.

A- Comparison between Proposed CTDHH Approach and Baseline Approaches

Table 5.17: Specification of scenarios for SIPHT SWFA in simulation environment

Scenario	No. of Tasks	No. of VMs
1	30	2
2	30	4
3	30	8
4	100	2
5	100	4
6	100	8
7	1000	2
8	1000	4
9	1000	8

This section provides a discussion about the completion time and total computational cost statistical results based on the comparison between proposed CTDHH approach and baseline approaches for each of the nine considered scenarios.

i. Completion time results:

Table 5.18 represents the descriptive statistical analysis for completion time results for SIPHT SWFA for all considered nine scenarios.

Table 5.18: Completion time comparison between CT-DHH and baselines for SIPHT

Scenario-1					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	3.48	3.48	3.48	3.48	3.348466667
MIN	3.48	3.48	3.48	3.48	2.71
MAX	3.48	3.48	3.48	3.48	3.658
S.D.	9.03362E-16	9.03362E-16	9.03362E-16	9.03362E-16	0.290713122
Scenario-2					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	7.95	8.17	7.342	7.764666667	6.505333333
MIN	6.74	6.74	3.7	3.7	3.7
MAX	10.04	10.04	10.04	10.04	10.04
S.D.	1.617437309	1.663222879	2.380723448	2.251728429	2.540197838
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	16.17333333	18.47973333	19.13333333	19.62666667	13.00073333
MIN	7.54	4.14	11.24	11.24	8.316
MAX	26.04	26.04	26.04	26.04	16.906
S.D.	6.893591894	7.378023055	4.632861135	5.226572249	1.97026289
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	10.87	10.87833333	10.87	10.87	10.6196
MIN	10.87	10.87	10.87	10.87	10.244
MAX	10.87	10.94	10.87	10.87	10.87
S.D.	3.61345E-15	0.021668877	3.61345E-15	3.61345E-15	0.311918822
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	25.35	26.27396667	26.15666667	29.05333333	20.87
MIN	20.3	13.134	11.2	20.3	11.2
MAX	30.4	83.499	30.4	30.4	30.4
S.D.	5.13633104	12.61821654	7.057482375	3.492033627	6.913263115

Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	49.93866667	49.83466667	48.53506667	47.73466667	38.48333333
MIN	11.64	11.64	17.58	11.64	26.7
MAX	73.6	73.6	73.6	73.6	63.1
S.D.	21.26467489	18.40849649	20.50696816	20.33561302	10.71438232
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	106.68	106.68	106.68	106.68	106.7043333
MIN	106.68	106.68	106.68	106.68	106.68
MAX	106.68	106.68	106.68	106.68	106.734
S.D.	2.89076E-14	2.89076E-14	2.89076E-14	2.89076E-14	0.025008045
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	241.056	246.6398667	244.0752667	225.5513333	170.6413333
MIN	106.68	60.04	78.674	106.68	106.68
MAX	290.8	290.928	290.828	290.8	193.9
S.D.	75.82318823	66.49934373	77.4065922	81.03379247	39.22950154
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	510.9004	549.2	503.98	523.36	423.23
MIN	97.488	290.8	290.8	193.9	290.8
MAX	678.4	678.4	678.4	678.4	581.5
S.D.	208.8424202	114.7477203	159.7305131	171.8391822	128.4599928

Table 5.18 shows that the average values of the completion time of the proposed CTDHH approach are very close to those of the other baseline approaches for most of the scenarios. This could be due to the nature of SIPHT SWFA having a less number of dependencies between their tasks, which gives smaller search space for the employed LLH algorithms to find an optimal solution especially when the number of VMs are less (scenarios one, four and six). However, in scenario 7 the average values of the completion time of the IWO approach is slightly lower than the proposed CTDHH approach, and this due to strong dependences between the tasks of SIPHT SWFA, the IWO approach managed to find the optimal with shorter convergence time.

Figure 5.26 illustrates the average completion time of CTDHH and baselines for SIPHT SWFA. This figure contains five charts and each of these charts is representing the relationship between the completion time and the considered scenarios for each of the approaches.

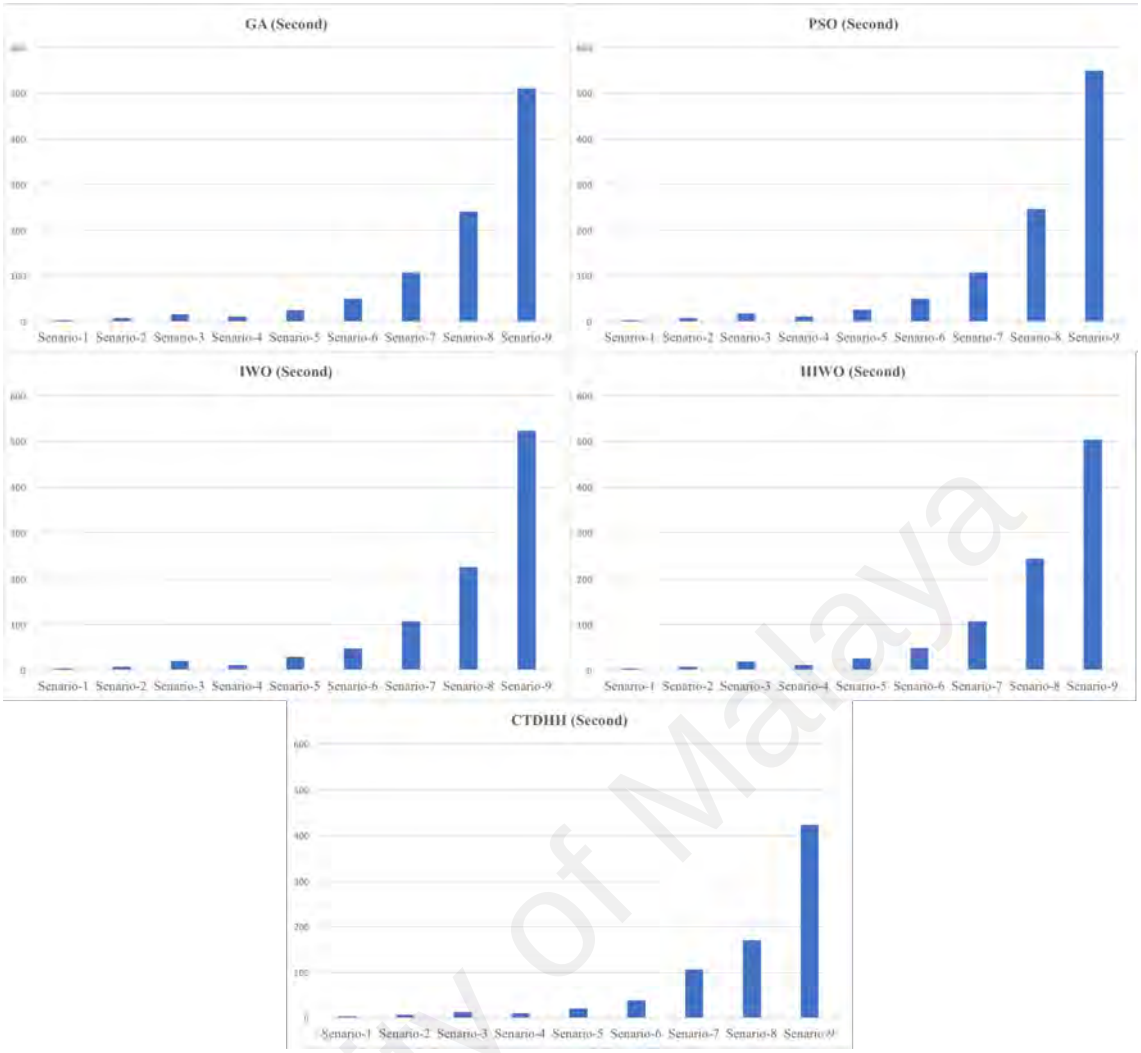

Figure 5.26: Average completion time of CTDHH and baselines for SIPHT SWFA

Figure 5.26 depicts that the average completion time (in second) of the proposed CTDHH approach is more optimal for all considered scenarios. However, the baseline meta-heuristic algorithms always attained very similar or closer values to each other, especially for scenarios one to seven.

Figure 5.27 shows the average completion time of all approaches for SIPHT SWFA of all considered scenarios.

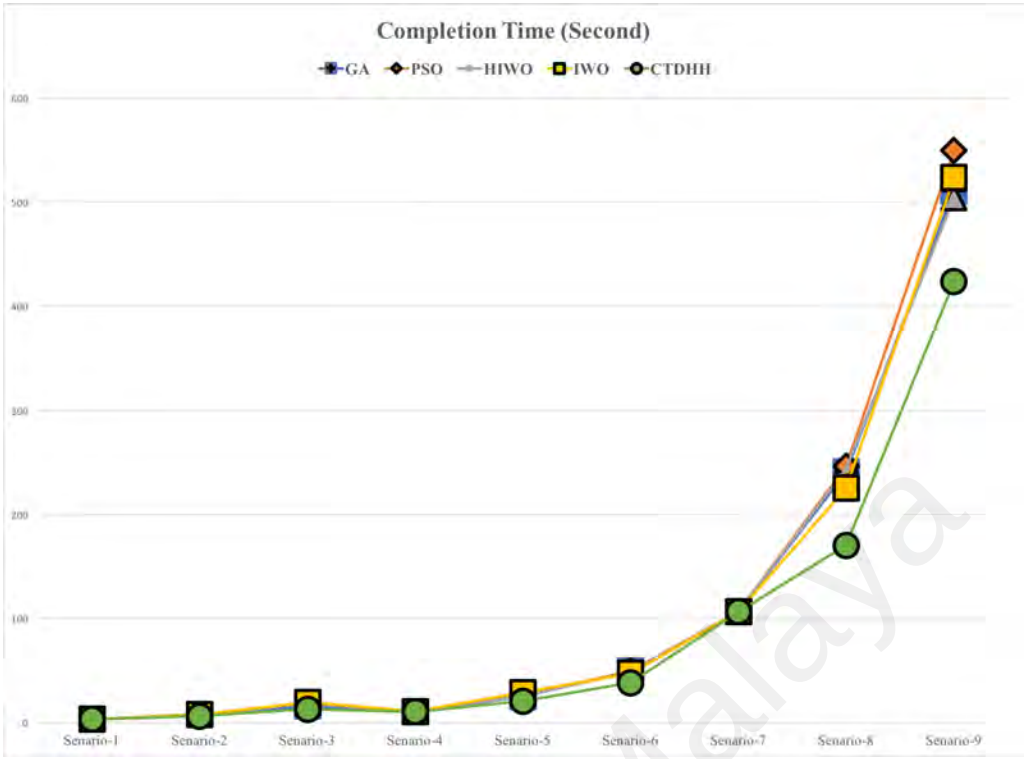

Figure 5.27: Average completion time of all approaches for SIPHT SWFA

Figure 5.27 summaries the results of Figure 5.26, where the average computation time values of the proposed CTDHH approach are much better for all scenarios.

ii. Total computational cost results

Table 5.19 represents the descriptive statistical analysis for total computational cost by comparing the proposed CTDHH approach and baseline approaches for SIPHT SWFA.

Table 5.19: Total computational cost comparison CT-DHH and baselines for SIPHT

Scenario-1					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.004084128	0.004084128	0.004084128	0.004084128	0.00392976
MIN	0.004084128	0.004084128	0.004084128	0.004084128	0.003180456
MAX	0.004084128	0.004084128	0.004084128	0.004084128	0.004293029
S.D.	1.86087E-18	1.86087E-18	1.86087E-18	1.86087E-18	0.000341181
Scenario-2					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.01866024	0.019176624	0.017233142	0.018225226	0.015269318
MIN	0.015820128	0.015820128	0.00868464	0.00868464	0.00868464
MAX	0.023565888	0.023565888	0.023565888	0.023565888	0.023565888
S.D.	0.003796449	0.003903917	0.005588034	0.005285257	0.005962352
Scenario-3					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.075924096	0.08675126	0.08981952	0.092135424	0.061030643
MIN	0.035395776	0.019434816	0.052765056	0.052765056	0.03903863
MAX	0.122242176	0.122242176	0.122242176	0.122242176	0.079363526

S.D.	0.032361278	0.034635391	0.021748503	0.024535621	0.009249202
Scenario-4					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.012757032	0.012766812	0.012757032	0.012757032	0.012463163
MIN	0.012757032	0.012757032	0.012757032	0.012757032	0.012022358
MAX	0.012757032	0.012839184	0.012757032	0.012757032	0.012757032
S.D.	2.48116E-18	2.54306E-05	2.48116E-18	2.48116E-18	0.000366068
Scenario-5					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.05950152	0.061670255	0.061394928	0.068193984	0.048986064
MIN	0.04764816	0.030828125	0.02628864	0.04764816	0.02628864
MAX	0.07135488	0.195988853	0.07135488	0.07135488	0.07135488
S.D.	0.012055996	0.029617478	0.016565323	0.008196501	0.016226811
Scenario-6					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.234432077	0.233943859	0.227843017	0.224085619	0.18065616
MIN	0.054642816	0.054642816	0.082527552	0.054642816	0.12534048
MAX	0.34550784	0.34550784	0.34550784	0.34550784	0.29621664
S.D.	0.09982489	0.086416846	0.096267911	0.095463502	0.050297596
Scenario-7					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.125199648	0.125199648	0.125199648	0.125199648	0.125228206
MIN	0.125199648	0.125199648	0.125199648	0.125199648	0.125199648
MAX	0.125199648	0.125199648	0.125199648	0.125199648	0.125263022
S.D.	1.98493E-17	1.98493E-17	1.98493E-17	1.98493E-17	2.93494E-05
Scenario-8					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	0.565806643	0.578913095	0.572893466	0.52941409	0.400529338
MIN	0.250399296	0.140925888	0.184663613	0.250399296	0.250399296
MAX	0.68256576	0.682866202	0.682631482	0.68256576	0.45512208
S.D.	0.177972187	0.15608726	0.181688753	0.190202518	0.092079486
Scenario-9					
	GA	**PSO**	**HIWO**	**IWO**	**CTDHH**
AVERAGE	2.398370838	2.57816448	2.365883712	2.456861184	1.986810912
MIN	0.457647667	1.36513152	1.36513152	0.91024416	1.36513152
MAX	3.18468096	3.18468096	3.18468096	3.18468096	2.7297936
S.D.	0.980389857	0.538671698	0.749838921	0.806681857	0.60304259

From Table 5.19, it is evident that the total computational cost values of the proposed CTDHH approach is showing much better results for all scenarios. This is due to fact that the proposed approach always consider finding the most optimal solution by choosing the suitable LLH algorithm for each iteration.

Figure 5.28 illustrates the average total computational cost of CTDHH and the baseline approaches for SIPHT. Figure 5.28 has five charts and each of these charts is representing the relationship between the total computational cost ($/hour) and the considered scenarios for each of the approaches.

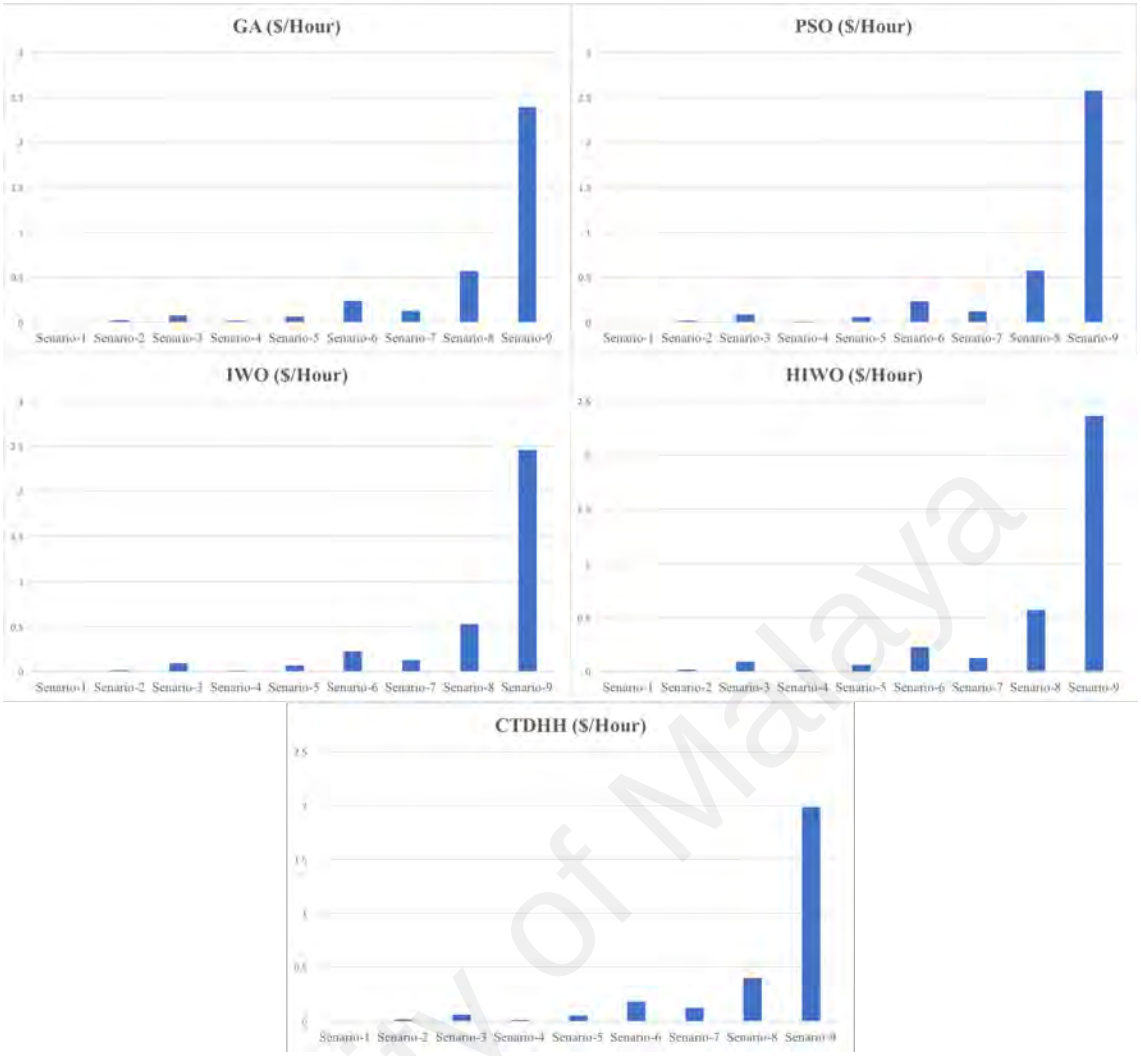

Figure 5.28: Average total computational cost of CTDHH and baselines for SIPHT

It can be observed from Figure 5.28 that the proposed CTDHH approach attains most optimal average total computational cost results for all considered scenarios and especially scenarios seven, eight and nine, where the number of tasks are high (1000 tasks). This shows the optimal performance of the proposed CHDHH with different types, complexities, and sizes of SWFAs.

Figure 5.29 illustrates the average total computational cost ($/hour) values of all approaches for the considered scenarios.

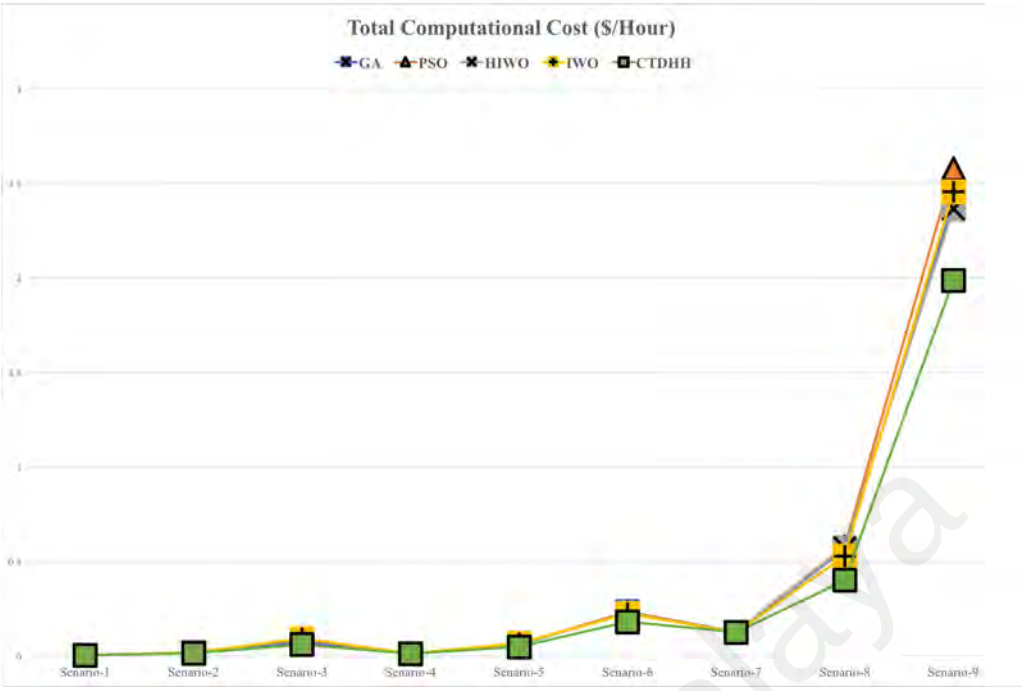

Figure 5.29: Average total computational of all approaches for SIPHT SWFA

From Figure 5.29, it can be concluded that the total computational cost results depend on the average completion values, which are ultimately affected by the size of datasets of the submitted SWFA as well as the number of available VMs.

B- Comparison between Proposed CTDHH Approach and HHSA Approach

In this section, the completion time and total computational cost statistical results have been discussed, which are based on the comparison between proposed CTDHH approach and HHSA approach for each of the nine considered scenarios.

i. Completion time results:

Table 5.20 illustrates the descriptive statistical analysis for completion time by comparing the proposed CTDHH approach and HHSA approach for SIPHT for all considered scenarios.

Table 5.20: Completion time comparison between CT-DHH and HHSA for SIPHT

Scenario-1		
	HHSA	**CTDHH**
AVERAGE	3.492	3.348466667
MIN	2.72	2.71
MAX	3.668	3.658
S.D.	0.182022356	0.290713122

Scenario-2		
	HHSA	**CTDHH**
AVERAGE	8.400766667	6.505333333
MIN	3.292	3.7
MAX	13.659	10.04
S.D.	2.044015853	2.540197838
Scenario-3		
	HHSA	**CTDHH**
AVERAGE	19.0858	13.00073333
MIN	9.79	8.316
MAX	26.152	16.906
S.D.	5.226268672	1.97026289
Scenario-4		
	HHSA	**CTDHH**
AVERAGE	10.89566667	10.6196
MIN	10.84	10.244
MAX	10.94	10.87
S.D.	0.03460948	0.311918822
Scenario-5		
	HHSA	**CTDHH**
AVERAGE	28.29933333	20.87
MIN	18.72	11.2
MAX	30.46	30.4
S.D.	4.18839442	6.913263115
Scenario-6		
	HHSA	**CTDHH**
AVERAGE	48.2526	38.48333333
MIN	20.67	26.7
MAX	73.6	63.1
S.D.	20.54516844	10.71438232
Scenario-7		
	HHSA	**CTDHH**
AVERAGE	106.7279333	106.7043333
MIN	106.68	106.68
MAX	106.734	106.734
S.D.	0.01644622	0.025008045
Scenario-8		
	HHSA	**CTDHH**
AVERAGE	253.0144	170.6413333
MIN	106.778	106.68
MAX	290.8	193.9
S.D.	62.99297419	39.22950154
Scenario-9		
	HHSA	**CTDHH**
AVERAGE	497.487	423.23
MIN	257.532	290.8
MAX	678.4	581.5
S.D.	177.6833579	128.4599928

It is evident from Table 5.20 that as much as the number of the submitted workflow tasks are getting bigger, the response of the proposed CTDHH approach still remains the same by achieving more optimal results than the HHSA approach. This can be seen in the results of scenarios one, four and nine, where the number of tasks are the highest (i.e., 1000 tasks). However, for the other smaller size scenarios, the results of the proposed CTDHH approach are higher but still close to the HHSA approach.

Figure 5.30 illustrates the average completion time of CTDHH and HHSA approach for

SIPHT SWFA. This figure contains two charts and each of these charts is representing the relationship between the completion time and the considered scenarios for each of the approaches. Note that y-axis denotes the completion time and x-axis denotes the considered scenarios.

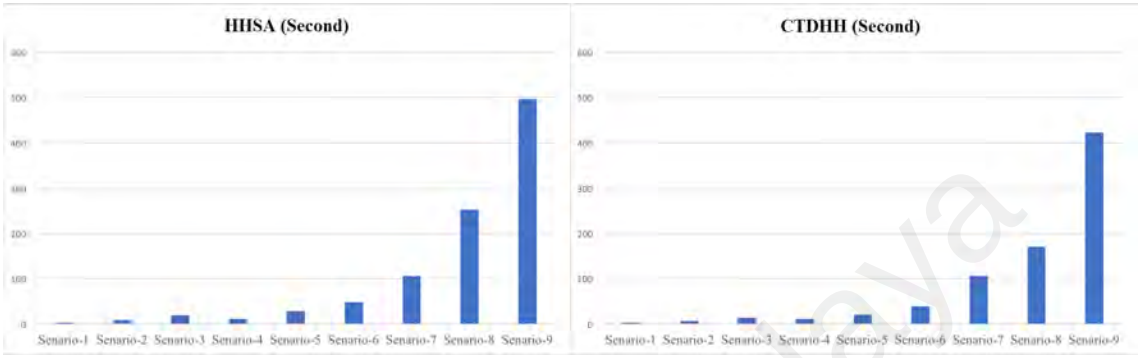

Figure 5.30: Average completion time of CTDHH and HHSA for SIPHT SWFA

From Figure 5.30, it can be seen that average completion time of the proposed CTDHH approach is much lower for all considered scenarios compared to the HHSA approach.

Figure 5.31 shows the average completion time of both approaches for SIPHT SWFA of all considered scenarios.

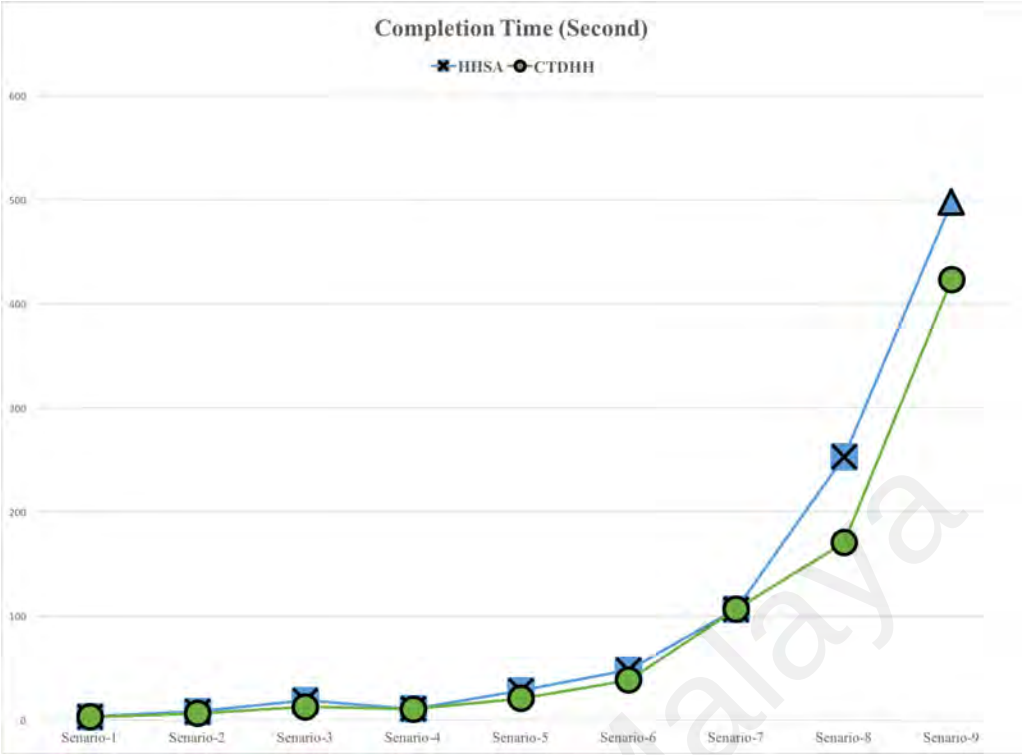

Figure 5.31: Average completion time of both approaches for SIPHT SWFA

From Figure 5.31, it is evident that the average completion time achieved by CTDHH and HHSA approach is significantly affected by the large numbers of workflow tasks in scenarios seven to nine. Furthermore, for scenarios one to six, the differences between the average completion time values are relatively small for both approaches.

ii. Total computational cost results

Table 5.21 represents the descriptive statistical analysis for total computational cost by comparing the proposed CTDHH approach and HHSA approach for SIPHT SWFA for all considered scenarios.

Table 5.21: Total computational cost comparison CT-DHH and HHSA for SIPHT

Scenario-1		
	HHSA	**CTDHH**
AVERAGE	0.004098211	0.00392976
MIN	0.003192192	0.003180456
MAX	0.004304765	0.004293029
S.D.	0.000213621	0.000341181
Scenario-2		
	HHSA	**CTDHH**

AVERAGE	0.01971828	0.015269318
MIN	0.007726982	0.00868464
MAX	0.032060405	0.023565888
S.D.	0.004797714	0.005962352
Scenario-3		
	HHSA	**CTDHH**
AVERAGE	0.08959638	0.061030643
MIN	0.045958176	0.03903863
MAX	0.122767949	0.079363526
S.D.	0.024534196	0.009249202
Scenario-4		
	HHSA	**CTDHH**
AVERAGE	0.012787154	0.012463163
MIN	0.012721824	0.012022358
MAX	0.012839184	0.012757032
S.D.	4.06177E-05	0.000366068
Scenario-5		
	HHSA	**CTDHH**
AVERAGE	0.066424195	0.048986064
MIN	0.043939584	0.02628864
MAX	0.071495712	0.07135488
S.D.	0.009830999	0.016226811
Scenario-6		
	HHSA	**CTDHH**
AVERAGE	0.226517005	0.18065616
MIN	0.097033248	0.12534048
MAX	0.34550784	0.29621664
S.D.	0.096447239	0.050297596
Scenario-7		
	HHSA	**CTDHH**
AVERAGE	0.125255903	0.125228206
MIN	0.125199648	0.125199648
MAX	0.125263022	0.125263022
S.D.	1.93013E-05	2.93494E-05
Scenario-8		
	HHSA	**CTDHH**
AVERAGE	0.5938754	0.400529338
MIN	0.250629322	0.250399296
MAX	0.68256576	0.45512208
S.D.	0.147857109	0.092079486
Scenario-9		
	HHSA	**CTDHH**
AVERAGE	2.335402973	1.986810912
MIN	1.208958221	1.36513152
MAX	3.18468096	2.7297936
S.D.	0.834116755	0.60304259

From Table 5.21, it can be seen that the total computational cost value of SWFS depends on the completion time (makespan) value and also on the VM available for each considered scenario. This table shows that the values of total computational cost of larger datasets (scenario seven, eight, and nine) are higher than those of the other smaller datasets (scenarios one to six). Logically speaking, larger size datasets require more time to execute than smaller size datasets, which significantly increases the overall computational cost. Also, the total computational cost of the proposed CTDHH approach has the lowest average value for most of the scenarios than the baseline approaches.

Figure 5.32 illustrates the average total computational cost of CTDHH and HHSA approach for SIPH SWFA. This figure depicts five charts, where each chart is representing the relationship between the total computational cost ($/hour) and the considered scenarios for each of the approaches.

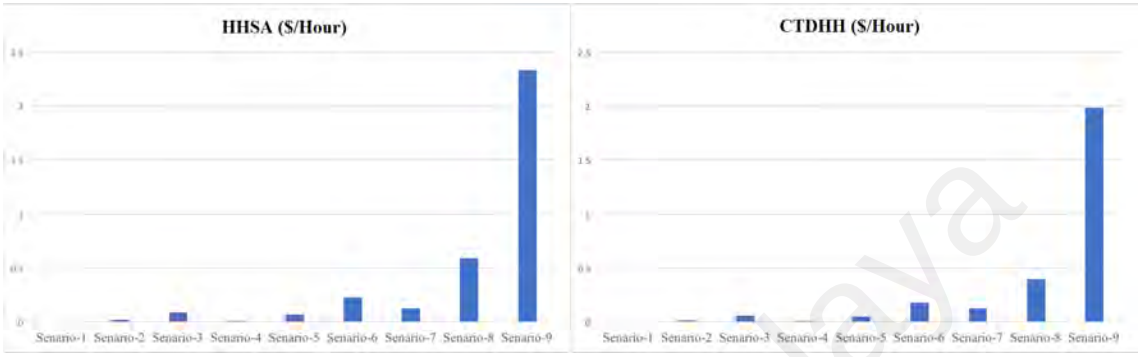

Figure 5.32: Average total computational cost of CTDHH and HHSA for SIPHT

Figure 5.32 shows that the average total computational cost of this application is much less than the Epigenomics and Inspiral SWFAs even using the same number of workflow tasks. This is due to the nature of considered dataset that has been executed by SIPHT SWFA. Furthermore, the proposed CTDHH approach always attains optimal performance results for all considered scenarios.

Figure 5.33 illustrates the average total computational cost ($/hour) values of both approaches for the considered scenarios.

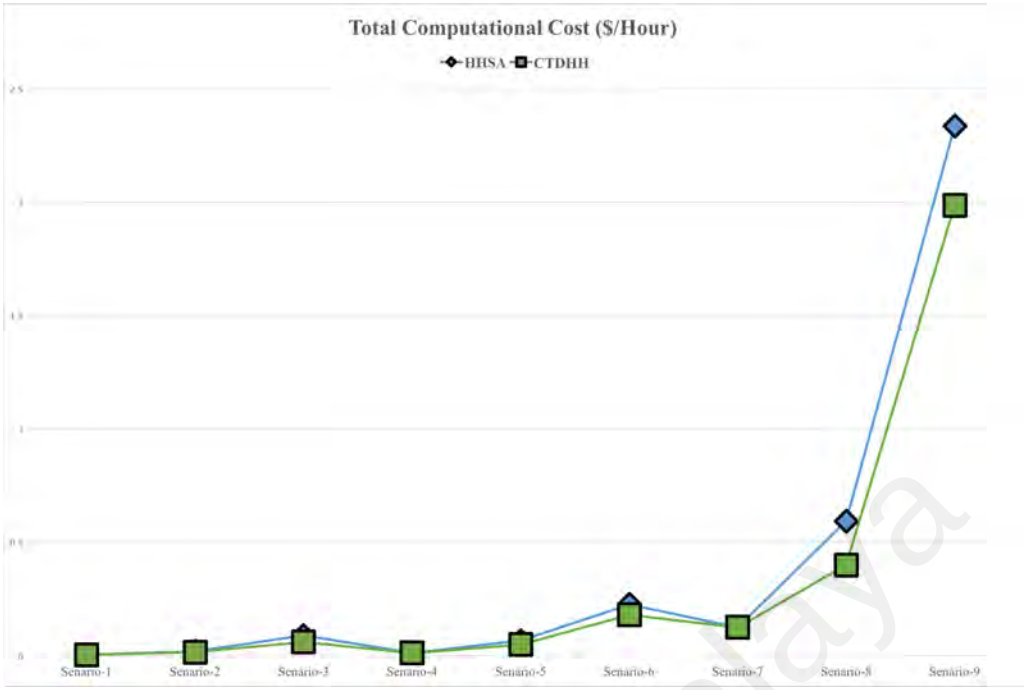

Figure 5.33: Average total computational of both approaches for SIPHT SWFA

From Figure 5.33, it is evident that there is a significant differences between the performance of SWFS approaches in scenarios eight to nine. This has confirmed that the cost results are dependent on the average completion values, which are ultimately affected by the size of dataset of the submitted SWFA as well as the number of available VMs. Moreover, it can be observed that for SIPHT SWFA, the HHSA approach does not show a good result comparing with previous SWFAs. This is due to the random behavior of the HHSA approach due to which HHSA cannot always get the optimal result.

5.4.2 Normality Test and Significance Test of the Proposed Approach

As it has been discussed in Section 3.3.5, this section presents the statistical results of normality test and significance test of the proposed approach.

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CTDHH	.113	30	.200*	.939	30	.088

Figure 5.34: Normality tests for the proposed CTDHH approach

Based on Figure 5.34, it can be concluded that results data is normally distributed, the paired T-test is used for statistical comparison of result data from the proposed CTDHH approach for each of the baseline approaches as well as the comparison of result data for the proposed CTDHH approach and HHSA approach.

As the result data of this research is considered as paired data, so there is a need to conduct statistical analysis tests to deal with such paired data. Therefore, a parametric test called Paired-Samples T-Test has been conducted to evaluate the significance between the proposed approach and the baseline approaches as well as the significance between the proposed CTDHH approach and the HHSA. As the total completion time result values are dependent on the completion time values, thus, it has been found that Paired-Samples T-Test value of both variables (i.e., completion time and total completion time) have the same T-Test values. The following tables (Tables 5.22 to 5.25) represent the T-Tests results of comparing the proposed approach and the baseline approaches as well as the significance between the proposed CTDHH approach and the HHSA approach.

Table 5.22: Normality and significance tests for Epigenomics SWFA

Scenario - 1		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-8.585	7.031E-10
CTDHH=PSO	-8.585	7.031E-10
CTDHH=HIWO	-8.585	7.031E-10
CTDHH=IWO	-8.585	7.031E-10
CTDHH=HHSA	-2.065	0.0238235
Scenario - 2		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-1.828	0.0389555

CTDHH=PSO	-3.027	0.0025725
CTDHH=HIWO	-2.208	0.01764
CTDHH=IWO	-2.162	0.0194905
CTDHH=HHSA	-1.828	0.0389555
	Scenario - 3	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-5.671	0.000002
CTDHH=PSO	-7.113	3.96485E-08
CTDHH=HIWO	-6.103	0.0000005
CTDHH=IWO	-4.765	0.0000245
CTDHH=HHSA	-2.369	0.012363
	Scenario - 4	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-8.008	3.92935E-09
CTDHH=PSO	-7.809	6.518E-09
CTDHH=HIWO	-8.008	3.92935E-09
CTDHH=IWO	-8.008	3.92935E-09
CTDHH=HHSA	-4.286	0.0000915
	Scenario - 5	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-8.474	1.2263E-09
CTDHH=PSO	-9.503	1.03645E-10
CTDHH=HIWO	-11.449	1.40555E-12
CTDHH=IWO	-9.187	2.1784E-10
CTDHH=HHSA	-3.279	0.0013555
	Scenario - 6	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.509	0.008977
CTDHH=PSO	-2.344	0.013071
CTDHH=HIWO	-3.218	0.0015855
CTDHH=IWO	-1.735	0.04664
CTDHH=HHSA	-0.621	0.2697515
	Scenario - 7	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-4.676	0.000031
CTDHH=PSO	-2.879	0.003714
CTDHH=HIWO	-4.676	0.000031
CTDHH=IWO	-4.676	0.000031
CTDHH=HHSA	-1.141	0.131515
	Scenario - 8	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-3.703	0.000445
CTDHH=PSO	-2.865	0.003837
CTDHH=HIWO	-2.955	0.0030755
CTDHH=IWO	-2.926	0.003303
CTDHH=HHSA	-5.923	0.000001
	Scenario - 9	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-6.436	2.4276E-07
CTDHH=PSO	-1.828	0.0389495
CTDHH=HIWO	-2.192	0.018291
CTDHH=IWO	-6.131	0.0000005
CTDHH=HHSA	-3.248	0.0014665

Table 5.23: Normality and significance tests for Inspiral SWFA

Scenario-1		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-3.031	0.0025415
CTDHH=PSO	-3.027	0.002572
CTDHH=HIWO	-3.031	0.0025415
CTDHH=IWO	-3.031	0.0025415
CTDHH=HHSA	-2.077	0.023405
Scenario-2		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.247	0.0162195
CTDHH=PSO	-6.041	0.0000005
CTDHH=HIWO	-3.108	0.0020985
CTDHH=IWO	-2.135	0.0206495
CTDHH=HHSA	-3.697	0.0004525
Scenario-3		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.882	0.00368
CTDHH=PSO	-2.491	0.0093515
CTDHH=HIWO	-2.363	0.012535
CTDHH=IWO	-2.07	0.0237355
CTDHH=HHSA	-3.729	0.000415
Scenario-4		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.409	0.0112975
CTDHH=PSO	-2.537	0.008414
CTDHH=HIWO	-2.409	0.0112975
CTDHH=IWO	-2.409	0.0112975
CTDHH=HHSA	-4.534	0.000046
Scenario-5		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-3.186	0.0017625
CTDHH=PSO	-3.131	0.0020255
CTDHH=HIWO	-2.635	0.0067845
CTDHH=IWO	-3.617	0.000581
CTDHH=HHSA	-5.771	0.0000015
Scenario-6		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.288	0.014793
CTDHH=PSO	-3.714	0.0004325
CTDHH=HIWO	-2.994	0.002792
CTDHH=IWO	-2.763	0.004925
CTDHH=HHSA	-2.684	0.0059505
Scenario-7		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-3.175	0.0017665
CTDHH=PSO	-3.362	0.001094
CTDHH=HIWO	-3.175	0.0017665
CTDHH=IWO	-3.175	0.0017665
CTDHH=HHSA	-3.281	0.0013475
Scenario-8		
	t-value	**Significat (1-tailed) test**

CTDHH=GA	-3.199	0.001662
CTDHH=PSO	-0.856	0.199423
CTDHH=HIWO	-0.506	0.308387
CTDHH=IWO	-2.551	0.00814
CTDHH=HHSA	-2.703	0.0056795
Scenario-9		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-1.416	0.0837295
CTDHH=PSO	-5.385	0.0000045
CTDHH=HIWO	-1.508	0.071241
CTDHH=IWO	-9.617	7.9515E-11
CTDHH=HHSA	-1.636	0.056306

Table 5.24: Normality and significance tests for Montage SWFA

Scenario 1		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-8.283	1.9665E-09
CTDHH=PSO	-8.283	1.9665E-09
CTDHH=HIWO	-8.283	1.9665E-09
CTDHH=IWO	-8.283	1.9665E-09
CTDHH=HHSA	-0.704	0.24359
Scenario 2		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.939	0.003201
CTDHH=PSO	-2.071	0.0236675
CTDHH=HIWO	-4.737	0.0000265
CTDHH=IWO	-4.516	0.0000485
CTDHH=HHSA	-2.939	0.003201
scenario 3		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-3.246	0.001476
CTDHH=PSO	-2.269	0.015458
CTDHH=HIWO	-3.807	0.0003365
CTDHH=IWO	-2.385	0.011927
CTDHH=HHSA	-3.12	0.002032
Scenario 4		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-1.232	0.113913
CTDHH=PSO	-3.226	0.0015525
CTDHH=HIWO	-5.882	0.000001
CTDHH=IWO	-1.62	0.058048
CTDHH=HHSA	-5.002	0.0000125
Scenario 5		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-3.306	0.001263
CTDHH=PSO	-3.768	0.0003745
CTDHH=HIWO	-2.795	0.0045575
CTDHH=IWO	-2.442	0.010465
CTDHH=HHSA	-2.107	0.0219375
Scenario 6		

	t-value	Significat (1-tailed) test
CTDHH=GA	-8.448	1.3057E-09
CTDHH=PSO	-8.448	1.3057E-09
CTDHH=HIWO	-8.448	1.3057E-09
CTDHH=IWO	-8.448	1.3057E-09
CTDHH=HHSA	-2.329	0.013518
	Scenario 7	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-4.573	0.0000415
CTDHH=PSO	-3.887	0.0002715
CTDHH=HIWO	-2.143	0.020302
CTDHH=IWO	-1.977	0.0288015
CTDHH=HHSA	-3.671	0.0004845
	Scenario 8	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-6.163	0.0000005
CTDHH=PSO	-2.343	0.013097
CTDHH=HIWO	-2.66	0.0062975
CTDHH=IWO	-2.288	0.014807
CTDHH=HHSA	-1.579	0.0625995
	Scenario 9	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-6.163	0.0000005
CTDHH=PSO	-2.343	0.013097
CTDHH=HIWO	-2.66	0.0062975
CTDHH=IWO	-2.288	0.014807
CTDHH=HHSA	-1.579	0.0625995

Table 5.25: Normality and significance tests for SIPHT SWFA

	Scenario 1	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.478	0.0096355
CTDHH=PSO	-2.478	0.0096355
CTDHH=HIWO	-2.478	0.0096355
CTDHH=IWO	-2.478	0.0096355
CTDHH=HHSA	-2.064	0.0240405
	Scenario 2	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-3.71	0.0004365
CTDHH=PSO	-3.156	0.0018565
CTDHH=HIWO	-1.145	0.130865
CTDHH=IWO	-2.031	0.0257605
CTDHH=HHSA	-3.408	0.0009695
	Scenario 3	
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.33	0.0134905
CTDHH=PSO	-4.095	0.0001545
CTDHH=HIWO	-6.994	5.4325E-08
CTDHH=IWO	-6.507	2.0045E-07
CTDHH=HHSA	-5.817	0.0000015
	Scenario 4	

	t-value	Significat (1-tailed) test
CTDHH=GA	-4.397	0.0000675
CTDHH=PSO	-4.573	0.0000415
CTDHH=HIWO	-4.397	0.0000675
CTDHH=IWO	-4.397	0.0000675
CTDHH=HHSA	-4.859	0.000019
Scenario 5		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.803	0.004469
CTDHH=PSO	-2.16	0.0195885
CTDHH=HIWO	-2.635	0.0066785
CTDHH=IWO	-5.027	0.000012
CTDHH=HHSA	-4.803	0.000022
Scenario 6		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.524	0.008676
CTDHH=PSO	-2.804	0.004457
CTDHH=HIWO	-2.417	0.0110825
CTDHH=IWO	-2.136	0.0206405
CTDHH=HHSA	-2.426	0.010858
Scenario 7		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	5.329	0.000005
CTDHH=PSO	5.329	0.000005
CTDHH=HIWO	5.329	0.000005
CTDHH=IWO	5.329	0.000005
CTDHH=HHSA	-4.509	0.0000495
Scenario 8		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-4.481	0.0000535
CTDHH=PSO	-4.833	0.00002
CTDHH=HIWO	-4.519	0.000048
CTDHH=IWO	-3.077	0.002268
CTDHH=HHSA	-7.183	3.30035E-08
Scenario 9		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-1.966	0.0294735
CTDHH=PSO	-3.719	0.0004265
CTDHH=HIWO	-2.171	0.019122
CTDHH=IWO	-2.868	0.0038145
CTDHH=HHSA	-1.956	0.0300615

From tables (Table 5.22 to 5.25) and based on the t-values and p-values, it can be observed that the proposed CTDHH approach has significant results for most of the Paired-Samples T-Test comparison results. This has ultimately confirmed that completion time and total computational cost results of the proposed CTDHH are more optimised compared to the baseline and existing approaches for most the SWFA datasets. However,

in the following exceptional cases, the proposed approach does not get significant results for the t-values and Paired-Samples T-Test (Table 5.22 to 5.25):

- In scenario 6 of Epigenomics SWFA the proposed CTDHH approach does not get significant results comparing with HHSA approach. This due to the fact that the hyper heuristic approches are always achieve optimal results.

- In scenario 8 of Inspiral SWFA the proposed CTDHH approach does not get significant results comparing with PSO and HIWO approaches. Also, in scenario 9 of the same SWFA, the proposed CTDHH approach does not get significant results comparing with HHSA approach. This due to the fact that the hyper heuristic approches are always achieve optimal results.

- In scenario 1 of Montage SWFA the proposed CTDHH approach does not get significant results comparing with HHSA approach. Also, in scenario 4 of the same SWFA, the proposed CTDHH approach does not get significant results comparing with GA and IWO approaches. Furthermore, in scenario 8 of the same SWFA, the proposed CTDHH approach does not get significant results comparing with HHSA approach. And finally, in scenario 9 of the same SWFA, the proposed CTDHH approach does not get significant results comparing with HHSA approach. Due to the high complexity of the Montage SWFA, the above-mentioned results shows that there is significant differences between the compared approaches.

- In scenario 2 of SIPHT SWFA the proposed CTDHH approach does not get significant results comparing with HIWO approach. This due to the fact that hybrid approch has achieved optimal results by considering the mutation operation of GA algorithm.

5.5 Summary

In this chapter, the statistical analysis methods have been extensively discussed, which were applied on the collected data by running the proposed approach and the baseline

approaches for cost optimisation of SWFS using WorkflowSim simulation environment. In order to efficiently evaluate the data-intensiveness and computational-intensiveness performance of the proposed approach, four scientific workflow application datasets (i.e., epigenomics, inspiral, montage, SIPHT) with different sizes of datasets have been executed using the simulation environment. All of descriptive analysis tests are particularly useful to get an observation of the data collected. Four statistical methods have been conducted in this chapter in order to perform complete analysis of the collected data. The four main values (i.e., average value, maximum value (represents the worst value), and minimum value (represents the best value), standard deviation)) have been selected for the completion time (makespan) and total computational cost.

For completion time results, it is concluded that the meta-heuristic algorithms (i.e., GA, PSO, IWO, and HWO) lack in achieving good results compared to the proposed approach. This is due to the nature of the solution proposed by these algorithms, which have limitations in considering for a more optimal solution, compared with the hyper-heuristic approach. It is worth mentioning that in some cases, the S.D. values of these algorithms are equal to zero because there is no variation to be measured. Also, the result has shown that the proposed approach has attained the most optimal values comparing with the other approaches. Besides this, the Epigenomics SWFA has always taken longer completion time (makespan) comparing with Inspiral, Montage, and Sipht SWFA, for all sizes of datasets. This is due to two reasons: (i) the size of tasks is large comparing with the other SWFAs, and (ii) the data dependencies (precedence constraints) between these tasks are more complex comparing with the other SWFAs. Furthermore, the results also show that the proposed approach has achieved the most optimal results for all types of SWFA datasets and for all considered sizes compared with the baseline approaches.

For total computational cost results, similar to the completion time results, the proposed

approach has achieved the cheapest total computational cost comparing with the baseline approaches. And these results are affected by the SWFA's type and size. This is mainly because of the complex and large size of the submitted workflow tasks, which ultimately make the SWFS approaches to take longer time to execute these tasks. Also, the Montage SWFA always consumes the lowest total cost compared with the other SWFA datasets (i.e., Epigenomics, Inspiral, and Sipht). This is due to the fact that the tasks of Montage SWFA have less precedence constraints compared to the other SWFA datasets.

CHAPTER 6: EVALUATION AND ANALYSIS USING REAL-WORLD ENVIRONMENT

In this chapter, the second type of the evaluation for the proposed Completion Time Driven Hyper-Heuristic (CTDHH) approach has been discussed, which is used to achieve the third objectives of this research. At the first stage, an extensive definition of the utilized Workflow Management System (WfMS) has been provided. At the second stage, the adopted settings and methodological steps for the conducted experiments have been comprehensively discussed. Finally, at the third stage, the results and statistical analysis have been provided by running the real-world based experiments using Pegasus WfMS. In the following sections (i.e. Section 6.1, Section 6.2 and Section 6.3) a complete detail for each of the above-mentioned stages has been provided.

6.1 Real-world Based Environment

The WfMS in cloud computing has the ability to handle the requests from different domains of SWFAs. To execute the SWFA datasets, high performance resources, such as supercomputers, need to be delivered by the service provider (i.e. infrastructure as a service) (Z. Wu et al., 2013; Yan et al., 2013; Deelman et al., 2013; Bittencourt & Madeira, 2013; Malawski et al., 2012). Therefore, WfMSs using cloud services enable the scientists to define multi-stage computational and data processing pipelines that can be executed as resources with predefined QoS. Therefore, the scheduling process can automate complex analyses, improves application performance, and reduces the time required to obtain the desired results (Sharif et al., 2013). Inspired by this, the studies that focused on cost optimisation of SWFS in cloud computing environment have been extensively surveyed.

Thanks to the above-mentioned importance of SWFAs, various WfMSs such as Pegasus (Deelman et al., 2005), Kepler (Ludscher, Altintas, Berkley, Higgins, & Zhao, 2006) and Taverna workbench (Wolstencroft, Haines, Fellows, Williams, & Fisher, 2013), Oozie

(Islam, Huang, Battisha, & Abdelnur, n.d.), ASKALON (Wieczorek, Prodan, & Fahringer, 2005), and GrADS (Berman, Casanova, Chien, & Cooper, 2005) have been successfully proposed for different types of scientific workflows. These WfMSs have been devised to define, manage, execute, and evaluate workflows in different computational environments besides cloud computing.

In this study, the Pegasus WfMS is utilized as a the second type of environment for evaluating the proposed approach.

This is mainly due to the fact that the use of Pegasus WfMS facilitated in explaining the complex computational tasks into workflows that link and manage sets of dependent tasks and related data files. The Pegasus WfMS can be defined as a configurable system for mapping and executing abstract application workflows over a wide range of execution environments including a laptop, a campus cluster, a Grid, or a commercial or academic cloud. Nowadays, Pegasus runs workflows on Amazon EC2, Nimbus, Open Science Grid, the TeraGrid, and many campus clusters. One workflow can run on a single system or across a heterogeneous set of resources. Pegasus can run workflows ranging from just a few computational tasks up to 1 million. Pegasus WfMS bridges the scientific domain and the execution environment by automatically mapping high-level workflow descriptions onto distributed resources. It automatically locates the necessary input data and computational resources necessary for workflow execution. Pegasus enables scientists to construct workflows in abstract terms without worrying about the details of the underlying execution environment or the particulars of the low-level specifications required by the middleware (Condor, Globus, or Amazon EC2). Pegasus WfMS also bridges the current cyber infrastructure by effectively coordinating multiple distributed resources. The input to Pegasus is a description of the abstract workflow in a XML format. Pegasus automatically chains dependent tasks together, so that a single scientist can complete

complex computations that once required many different people. Pegasus has a number of features that contribute to its usability and effectiveness (Deelman et al., 2005; Deelman, Blythe, Gil, & Kesselman, n.d.; Deelman, Vahi, Juve, Rynge, & Livny, 2015).

1. *Portability/Reuse* : User-created workflows can easily be run in different environments without any alteration. Pegasus currently runs workflows on top of Condor, Cloud infrastructures such as Open Science Grid and TeraGrid, Amazon EC2, and Nimbus. The same workflow can run on a single system or across a heterogeneous set of resources.

2. *Performance* : The Pegasus mapper can reorder, group, and prioritise tasks in order to increase the overall workflow performance.

3. *Scalability* : Pegasus can easily scale both the size of the workflow tasks as well as the distributed computational resources. Pegasus runs workflows ranging from just a few computational tasks up to 1 million. The number of resources involved in executing a workflow can scale as needed without any impediments to performance.

4. *Provenance* : By default, all jobs in Pegasus are launched via the kick-start process that captures runtime provenance of the job and helps in debugging. The provenance data is collected in a database, and the data can be summaried with tools such as Pegasus-statistics, Pegasus-plots, or directly with SQL queries.

5. *Data Management* : Pegasus handles replica selection, data transfers and output registrations in data catalogs. These tasks are added to a workflow as auxiliary jobs by the Pegasus planner.

6. *Operating Environments* : Pegasus workflows can be deployed across a variety of computational environments such as Cloud, Local Execution, Grid, Condor Pools and Glideins. The Cloud computing environment uses a network as a means to connect a Pegasus end user to distributed resources that are based in the cloud.

6.2 Experimentation Setting

In order to implement and develop the proposed CHDHH approach in the real-world experimentation environment, four main phases have been performed. In the first phase, the experimentation test-bed layers has been planned. At the second phase, configuring and testing the experimentation environment in order to prepare for the implementation phase. At the third phase, the proposed CTDHH approach has been implemented. And finally at the fourth phase, the experiments have been conducted to evaluate the performance of the proposed CTDHH approach. Figure 6.1, represents the main phases of the considered experimentation.

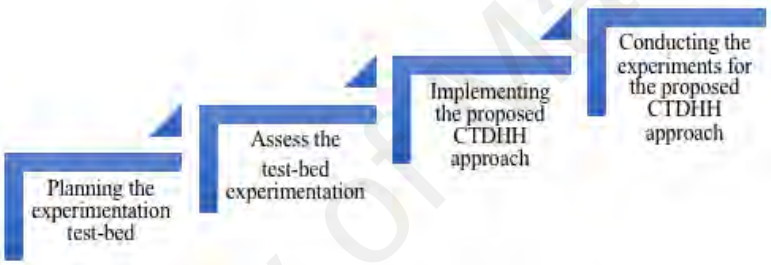

Figure 6.1: Phases of test-bed experimentation

The following sections provide a complete detail about each of these phases.

6.2.1 Planning the Experimentation Test-bed

As mentioned earlier, several types of frameworks have been considered by the researchers in the literature. For this reason, planning and selecting the right real-world test-bed tools is one of the most challenging researchers usually face. In this section, the main layers have been listed (i.e. SWFA layer, Pegasus layer, HTCondor layer, Visualization layer, and Physical layer) that have been considered to implement the SWFS test-bed experimentation.

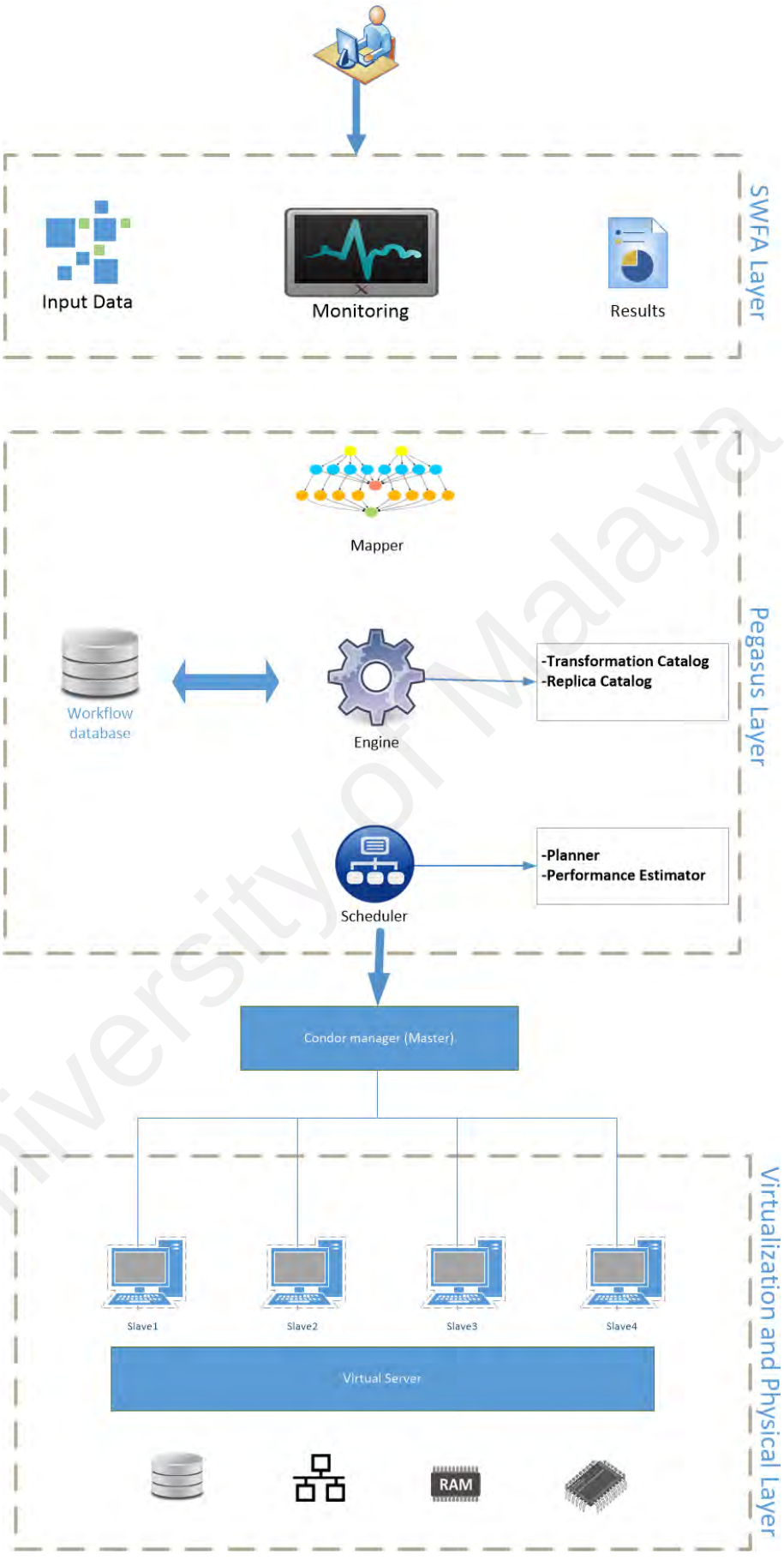

Figure 6.2: Experimentation test-bed layers

The following are the explanation of these layers (Figure 6.2).

6.2.1.1 SWFA layer

For Scientific Workflow Application (SWFA) layer, the usage of real-world SWFA can be efficiently utilised to evaluate the performance of the proposed CTDHH approach. The main reason behind choosing the real-world SWFA is that the real dependent tasks and data will be executed in real computational resources, while collecting accurate results for the considered completion time (makespan) and total computational cost parameters.

Based on the literature review and discussions with the experts in this area of research, the Montage application has been selected as the SWFA dataset. Three different sizes have been considered in order to evaluate the data-intensiveness of the proposed approach. Montage application has been developed on the top of Pegasus WfMS. The utilised package of Montage application has several types of files, for instance, the tasks (executable programming files), data (set of images that need to be processed) and others scripts, which are used to facilitate the configuration of Pegasus WfMS. The scripts of Monatge SWFA have been modified in order to establish the communication between this application and HTCondor pool, which ultimately supports the sending and receiving of tasks and images from Montage to the HTCondor pool. Furthermore, the Montage SWFA in real-world datasets is a well-known SWFA in Pegasus due to its tasks and data complexity. Montage (http://montage.ipac.caltech.edu) is a SWFA for assembling Flexible Image Transport System (FITS) images into custom mosaics. The main files component is the set of dependent programs (tasks) and sets of images (data). With the help of Pegasus, these files will be submitted and executed in an automated way. There are several features that scientists can benefit from using the Montage SWFA in real world as. More details for the Montage SWFA are available in Appendix B.3.

6.2.1.2 Pegasus layer

As it has been mentioned earlier in Section 6.1, Pegasus layer as the core layer of the considered test-bed experimentation is considered as the WfMS of experimentation setup. The Pegasus WfMS (Version 4.6) was installed using Github repository hosting service on the Ubuntu Linux operating system (Version 16.04 LTS). A Pegasus WfMS defines, manages, and executes workflows on available virtual machines (i.e. computing resources), where the workflow execution order is driven by a computer representation of the workflow logic. There are several stages that have been performed by the Pegasus WfMS in order to accomplish the submitted workflow tasks. Each stage is responsible to process the submitted tasks based on the different underlying techniques. For instance, modeling stage can be done using Directed Acyclic Graph (DAG) technique to highlight the precedence constraints of the submitted workflow tasks. In comparison to the modeling stage, scheduling or planning is considered as a core workflow processing stage of a WfMS. After processing the workflow tasks in modeling and scheduling stages, WfMS needs to submit the scheduled tasks to the execution stage using the HTCondor pool.

6.2.1.3 HTCondor layer

The execution process of Pegasus WfMS can be applied using various computational environments. For instance, previously researchers have used clustering computing, parallel computing, and grid computing. After the emergence of cloud computing, researchers have started migrating their WfMS to the cloud computing environment as it offers more powerful, scalable, flexible, and virtualised features. The main goal of using the condor pool is to implement, develop, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources. Furthermore, a HTCondor is a compute-intensive cloud system that provides a mechanism called DAGMan to schedule workflows within the HTCondor enabled grid.

DAGMan is a meta-scheduler that manages the workflow tasks dependencies. Its name is derived from the structure of the workflows (DAG) that it schedules. DAGMan relies on the workflow description to submit tasks in a predefined order. However, DAGMan is limited in terms of the task dependencies. In particular, it is unable to handle branching or looping. The actual scheduling of the tasks is done by the Condor scheduler. Based on the recommendations from the literature, one manager node and eight slaves (workers) has been configured, where each of these nodes represents a virtual machine. The specification of these VMs are (1 processor - intel Xeon CPU E5540 @ 2.53GHz, 4 GB RAM, 100GB Hard disk for Master and 10 GB Hard disk for each of the slaves).

6.2.1.4 Virtualization layer

All the VMs were installed on EXSi VMware Virtual Server 5.5, which is configured on VSphere Client 5.5.

6.2.1.5 Physical layer

As the physical layer of the test-bed, the EXSi VMware Virtual Server 5.5 has been configured on the physical server: *Intel(R) Xeon(R) CPU E5-24070 @ 2.20GHz.*

6.2.2 Assessing the Test-bed Experimentation

In order to investigate the test-bed experimentation, the already implemented workflow examples (e.g., hello world example) in Pegasus have been implemented. This testing step has helped to check the data-flow of the workflow tasks submitted to the SWFA side through the Pegasus WfMS to the available VM slaves of the condor pool.

6.2.3 Implementing the Proposed CTDHH Approach

In this section, the considered implementation stages of the proposed CTDHH approach is described. Figure 6.3 illustrates the main steps that have been considered in implementing

the proposed approach, starting from the submission step and finishing at collecting results. In order to automate the proposed CTDHH approach, two main dashboards (web-interfaces) have been implemented, (i) Run dashboard, and (ii) Monitoring dashboard. Both web-based dashboards are used to simplify the task for the user (i.e. scientists) to manage the submitted workflow tasks to WfMS in an automated manner. The following section elaborates on the steps for each of these dashboards:

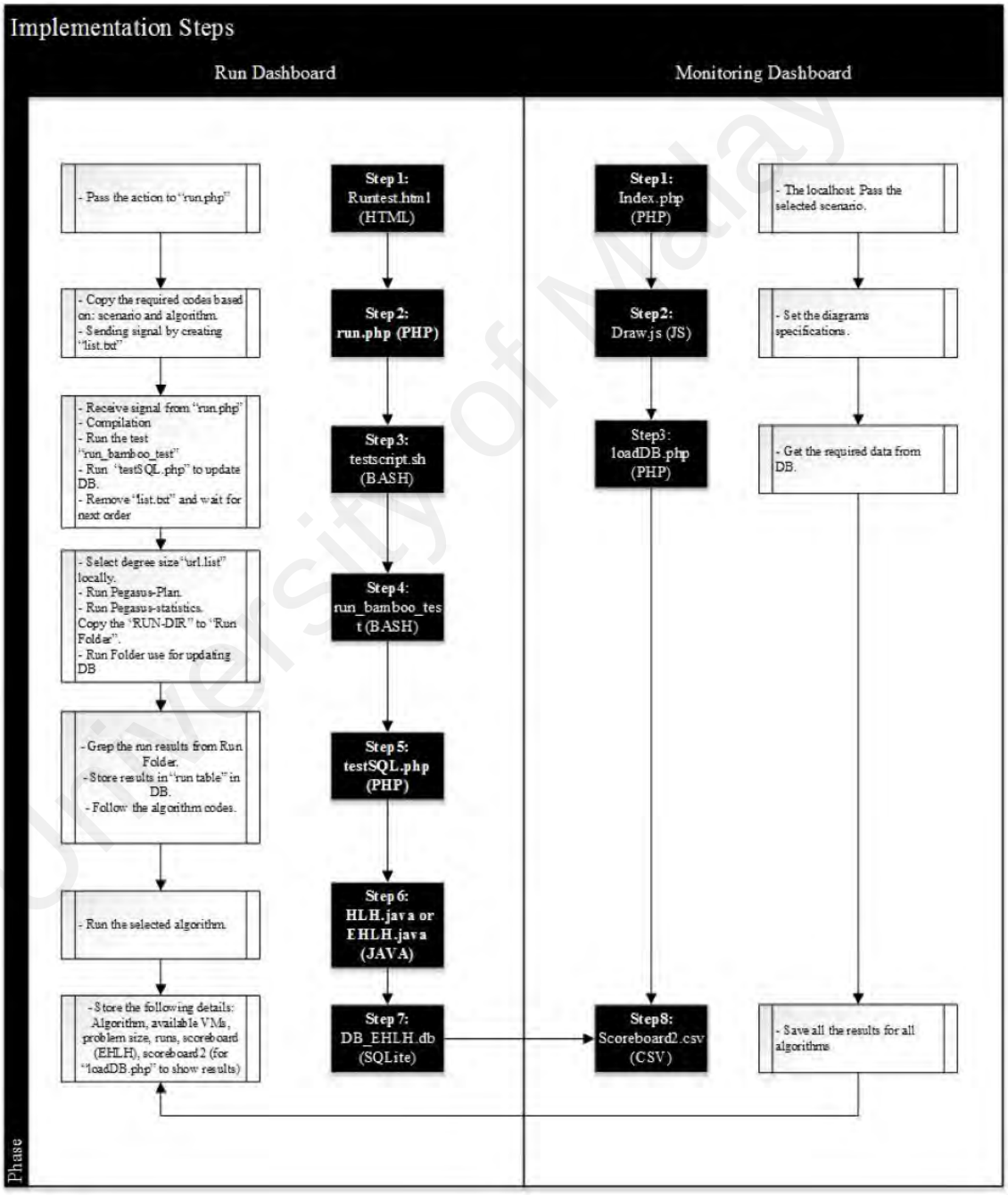

Figure 6.3: Implementing steps of the proposed CTDHH approach

6.2.3.1 Run Dashboard

Figure 6.4 shows the print-screen of the implemented run dashboard, in this page, the user can select the required scenario by choosing the targeted SWFS approach as well as the number of available VMs and size of workflow tasks (degree). After selectng the proposed CTDHH approach and scenario type details, the selected approach passes these details from the HTML page to PHP code “run.php” to send a signal to Linux O.S. (Version 16.04 LTS) by creating “list.txt” file. Next, BASH file in Linux O.S. will receive the details from “run.php” and do the compilation then run the experimentation based on the selected scenario. After the compilation is done successfully, the BASH file “testscript.sh” runs the experiment and updates the database by running the database updater “testSQL.php”. The other task of this BASH file is to remove the “list.txt” once the task is done and wait for the next order send by “run.php” file. To run the experimentation, the “run-bamboo-test” BASH file is used to pass the required details to Pegasus environment, by selecting the targeted list of datasets (images) and then run the “Pegasus-run” command. After the run is completed successfully, the approach will run “Pegasus-Statistics” command to get all the required statistical results for that particular run. In order to keep all the achieved runs’ results, after each run, the complete results will be saved in an archived folder called “Run-Folder”. This folder will be used to update the SQL database tables using testSQL.php script which will grip the run results from all the relevant files in "Run-Folder” and save them in a run table as a database file. After this step is done, the proposed approach will follow the other steps using the java class implemented in Pegasus (EHLH.java). Once all the steps of the proposed CTDHH approach is done successfully, the approach will run “loadDB.php” to store the following results in SQL database: algorithm (the used approach name), available VMs (number of VMs), and problem size (degree size). It is worth to mention that the "scoreboard table" (see Chapter 4 for complete details about the

run table and scorebored table) is used by the proposed CTDHH approach to update the Time Score (TS) after each run and also it will be used for the next run. The results of all the runs can be imported by the user using "CSV" format to be used by the Monitoring Dashboard to view the required results based on the user selection.

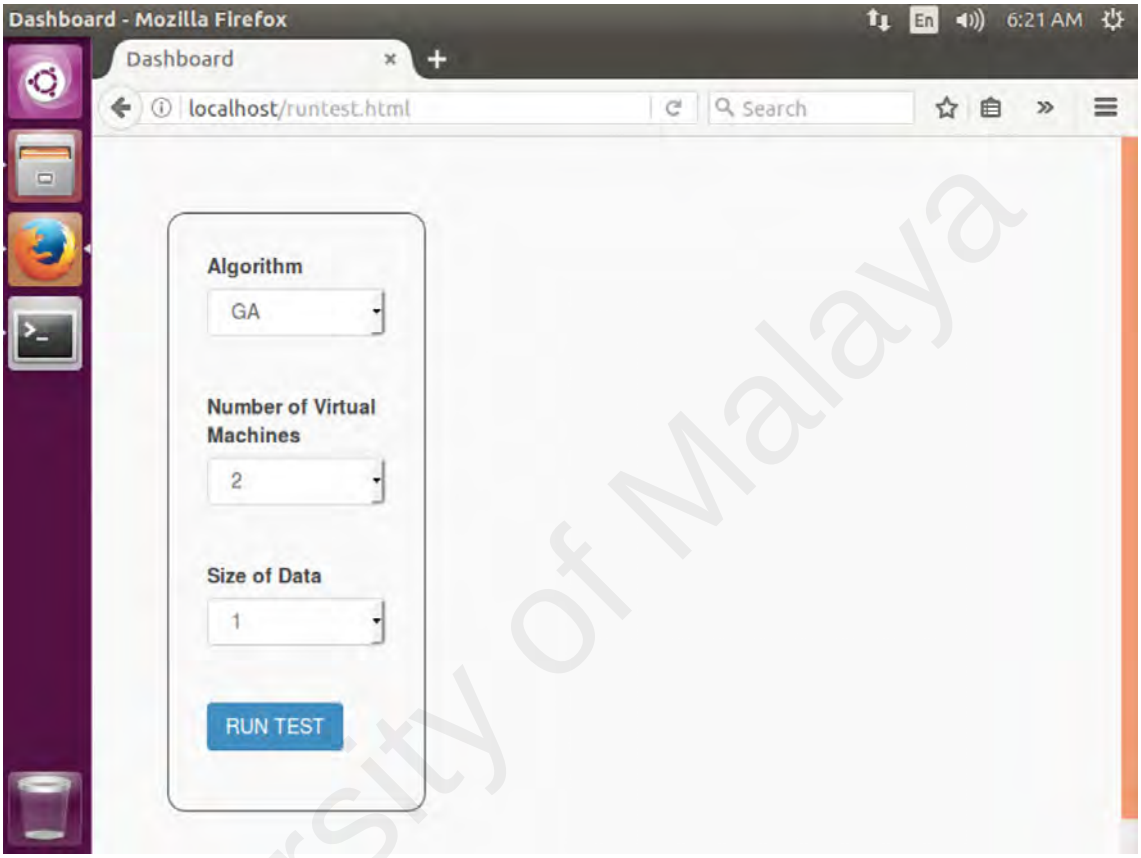

Figure 6.4: Run dashboard of the proposed CTDHH approach

6.2.3.2 Monitoring Dashboard

As shown in Figure 6.5, the monitoring dashboard page can be used by the user to display and compare the results of the completion time and total computational of the proposed CTDHH approach with the baseline approaches and HHSA approach. At the first step, the user can select the targeted scenario by selecting the degree size and number of VMs. The approach will automatically retrieve the scenario details and pass them to JavaScript file called "Draw.js" to set the diagrams specifications. The "loadDB.php" file will be used to get the diagrams details from the database to display them in "index.php" page. The

"index.php" page shows the results of the completion time and total computational cost for the user to be able to know the performance of the proposed approach as well as the other baseline approaches and HHSA approach.

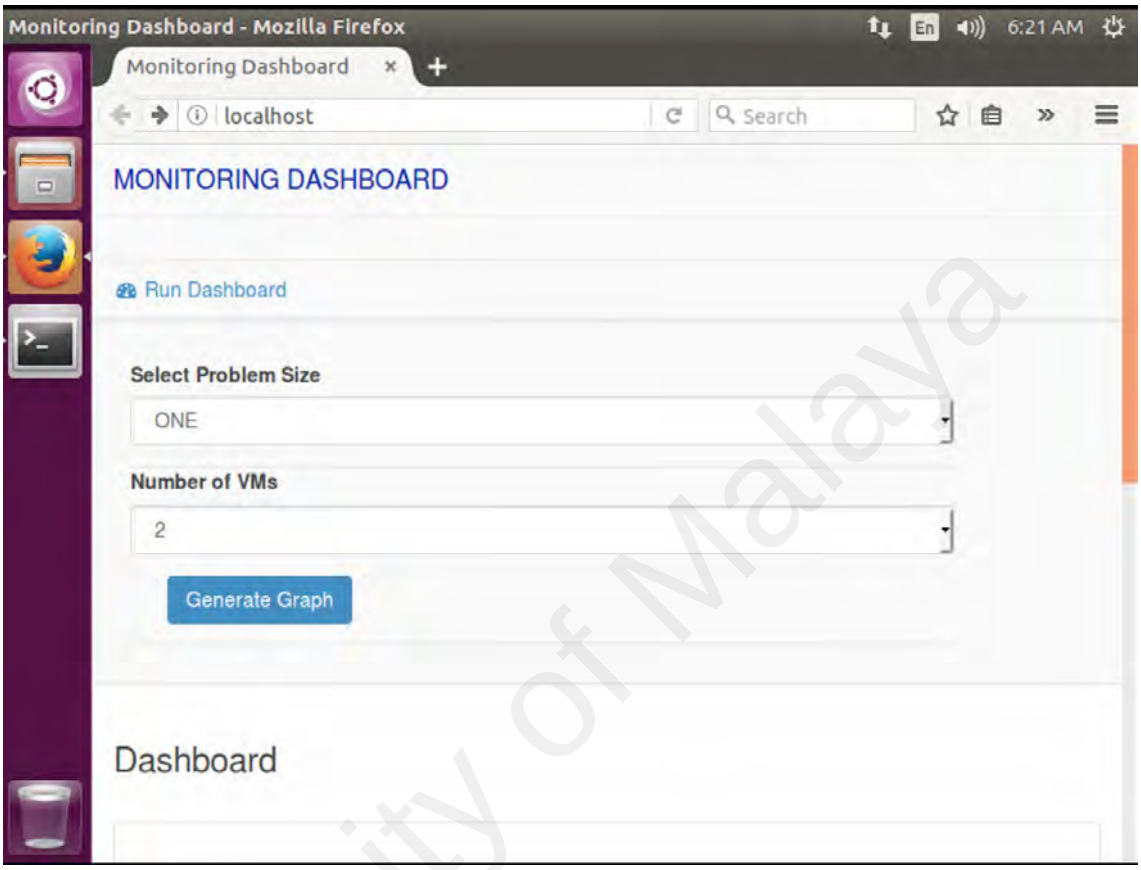

Figure 6.5: Monitoring dashboard of the proposed CTDHH approach

6.2.4 Conducting the Experiments for the Proposed CTDHH Approach

The first step of the experimentation is selecting the proposed CTDHH approach using the "Run Dashboard". Then, compile the Pegasus using "ant" compiler to update the files. The next step is choosing the size of the Montage SWFA and finally by running the Montage SWFA in the real environment. In order to comprehensively evaluate the proposed approach, nine experimentation scenarios have been considered by running the proposed CTDHH approach and baseline as well as HHSA approach using different degree sizes of Montage SWFA by utilising the master node and other eight slaves' nodes of the condor pool. Table 6.1 presents each of the considered scenarios, where each of the

Table 6.1: Specification of scenarios for Montage SWFA in real-world environment

Scenario	Size of Workflow	No. of Tasks	No. of Jobs	No. of VMs
1	Degree-1	387	44	2
2	Degree-1	387	44	4
3	Degree-1	387	44	8
4	Degree-2	1442	95	2
5	Degree-2	1442	95	4
6	Degree-2	1442	95	8
7	Degree-3	2425	150	2
8	Degree-3	2425	150	4
9	Degree-3	2425	150	8

considered scenarios has a different number of workflow tasks and a different number of VMs.

It is worth to mention that the "pegasus-statistics" and "pegasus-analyser" command of Pergasus WfMS have been used to check the status of the submitted workflow tasks and the statistical results from running each approach. Based on the literature and because of population search based approaches that have been considered in this study for the conducted experimentation scenarios, each approach has been executed for 30 times.

6.3 Results and Discussion of Real-world Environment

Similar to the simulation environment for the real-world evaluation environment, the proposed approach Completion Time Driven Hyper-Heuristic (CTDHH) approach has been compared with four baseline meta-heuristic approaches (i.e Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Invasive Weed Optimisation (IWO), and Hybrid Invasive Weed Optimisation (HIWO)) and an existing hyper-heuristic approach named Hyper-Heuristic Scheduling Algorithm (HHSA). The complete details for each of the used baseline approaches and HHSA approach is provided in Appendix A.

There are nine scenarios that have been utilized in the experiments and each of these scenarios has a different number of workflow tasks and a different number of VMs. Two

main types of statistical analysis have been used in this chapter to evaluate the performance of the proposed CTDHH approach in the real-world environment including (i) descriptive statistical analysis, and (ii) Normality testing and significant statistical analysis.

6.3.1 Descriptive Statistical Analysis

Four types of descriptive statistical analysis have been used for the experimentation (average, minimum, maximum, and standard deviation (S.D.)) to compare the proposed CTDHH approach with the baseline and HHSA approaches based on the completion time and total computational cost values. The minimum statistical value represents the best value and the maximum statistical value represents the worst result. In the following sections, the experimentation results of the descriptive statistical analysis of the considered Montage SWFAs in the real-world environment have been presented and discussed.

6.3.1.1 Comparison between Proposed CTDHH Approach and Baseline Approaches

In the this section, the completion time and total computational cost statistical results have been discussed on the comparison between proposed CTDHH approach and baseline approaches for each of the nine considered scenarios.

i. Completion time results

Table 6.2 represents the descriptive statistical analysis of completion time results for Montage SWFA in the real-world environment for all considered nine scenarios.

Table 6.2: Completion time comparison between CTDHH and baselines for Montage

Scenario-1					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	380.5	378.9	373.5333333	392.1	370.9333333
MIN	344	351	352	370	349
MAX	401	419	399	446	396
S.D.	15.4378621	16.655433	11.55417027	15.34960406	14.64271817
Scenario-2					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	364.5333333	354.2333333	356.5333333	379.5333333	344.6
MIN	332	333	337	349	326
MAX	395	373	374	423	382
S.D.	13.10628523	9.394361234	8.443456284	18.30909579	13.58142748

Scenario-3					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	363.1333333	351.8333333	367.5	479.4666667	342.1666667
MIN	329	332	333	370	321
MAX	389	368	440	805	378
S.D.	15.92381286	9.627845153	22.23502983	89.03998843	14.57935085
Scenario-4					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	1336.9	1389.366667	1061.6	1104.633333	1065.533333
MIN	1252	1330	1016	1040	1004
MAX	1390	1466	1145	1237	1114
S.D.	28.1674182	29.1139063	28.70371837	43.41975026	31.40144224
Scenario-5					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	1085.766667	1099.266667	1180.566667	1159.466667	1027.1
MIN	1024	1032	1088	1078	976
MAX	1260	1208	1478	1461	1145
S.D.	59.47742738	43.7161166	108.0863905	69.23758862	38.27878678
Scenario-6					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	982.9	1007.3	1014.333333	963.3333333	956.5
MIN	915	955	937	919	909
MAX	1200	1062	1085	1041	1023
S.D.	79.02437424	27.79251946	37.38461175	32.31134458	27.5690356
Scenario-7					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	1889.966667	1844.633333	1819.7	1755	1706.966667
MIN	1757	1778	1610	1675	1627
MAX	2273	1958	2099	1843	1795
S.D.	108.4522725	43.92507989	168.2834534	49.62827337	46.0033607
Scenario-8					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	1746.633333	1712.2	1654.333333	1674.766667	1635.566667
MIN	1639	1610	1578	1593	1576
MAX	1905	2014	1817	1881	1711
S.D.	52.78550175	99.94874549	54.5845332	70.28399697	35.0818583
Scenario-9					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	1856.833333	1948.7	1920.6	1987.466667	1715.866667
MIN	1747	1750	1818	1885	1622
MAX	2068	2694	2079	2162	1864
S.D.	85.08092834	276.1708804	62.11924018	62.94263033	72.47674221

It can be observed from Table 6.2 that the proposed CTDHH approach has the most optimal average completion time (second) compared with the other baseline approaches. Regarding the S.D. value, HIWO meta-heuristic algorithm achieved better results comparing to the proposed CTDHH approach. This is due to the reason that after each run, selection operator of CTDHH approach is required to select most suitable LLH from a list of LLHs, which ultimately has affect the S.D. However, the average value of the completion time is always considered by researchers for scheduling problem (Abrishami & Naghibzadeh, 2012; Saeid Abrishami, 2013).

Figure 6.6 presents five charts. Note that each of these charts is representing the

relationship between the average completion time and the considered scenarios for each of the approaches.

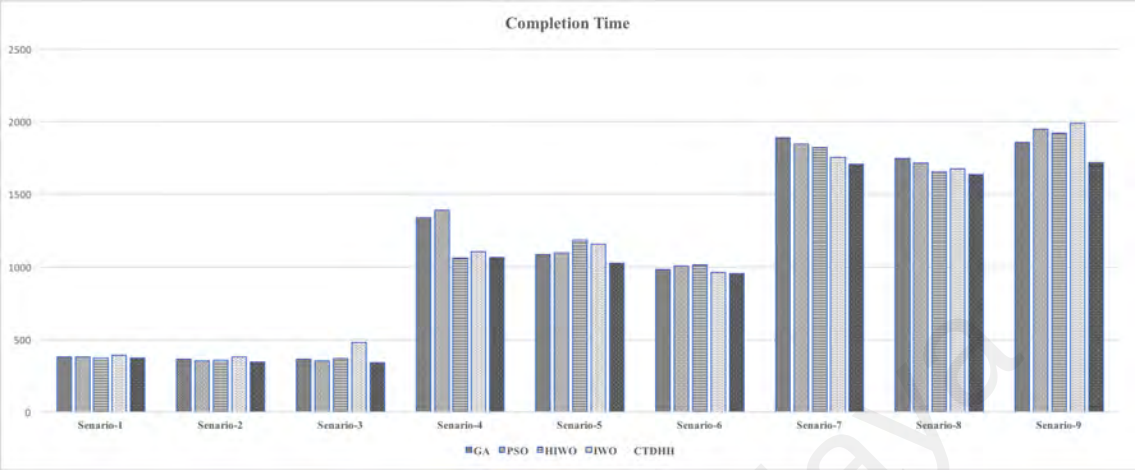

Figure 6.6: Average completion time of CTDHH and baselines for Montage in real environment

The first observation from these charts (Figure 6.6) is that the proposed CTDHH approach achieved the most optimal average completion time values. Also, the average completion time values of all approaches are affected by the size of dataset (degree) as well as the availability of VMs. It can be also observed that there are slight changes of the average value for scenarios one, two and three of all approaches. This is because of the limited number of available VMs and the problem size is smaller comparing with other scenarios (scenarios four to six). On the other hand, the average completion time of the proposed approach has achieved the most optimal value comparing with all other baseline approaches. This is due to the fact that the HLH strategy of the proposed CTDHH approach depends on the TS of the execution results of employed LLH algorithms.

Figure 6.7 shows the average completion time of all approaches for the considered scenarios.

Figure 6.7: Average completion time of all approaches for Montage in real environment

It can be observed from Figure 6.7 that the values of average completion time of scenario one to scenario three are always lower than the average completion time of all other scenarios. Also, the other observation is that the average completion time values of scenario seven to scenario nine are always higher than the average completion time of all other scenarios. The proposed CTDHH approach has achieved the lowest completion time performance comparing with the performance of all other baseline approaches and the performance of the proposed CTDHH approach is getting better as the size of dataset increases.

ii. Total computational cost results

Table 6.3 presents the statistical results of total computational cost for Montage SWFA in the real-world environment for all considered scenarios.

Table 6.3: Total computational cost comparison CTDHH and baselines for Montage

Scenario-1					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	1.7506044	1.74324312	1.71855216	1.80397368	1.70659008
MIN	1.5826752	1.6148808	1.6194816	1.702296	1.6056792
MAX	1.8449208	1.9277352	1.8357192	2.0519568	1.8219168
S.D.	0.07102652	0.07662832	0.05315843	0.07062046	0.06736822
Scenario-2					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	3.35428992	3.25951344	3.28067712	3.49231392	3.17087136
MIN	3.0549312	3.0641328	3.1009392	3.2113584	2.9997216
MAX	3.634632	3.4321968	3.4413984	3.8922768	3.5150112
S.D.	0.12059879	0.08644315	0.07769331	0.16847298	0.12497086
Scenario-3					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	6.68281536	6.4748592	6.763176	8.82372096	6.2969616
MIN	6.0546528	6.1098624	6.1282656	6.809184	5.9074272
MAX	7.1588448	6.7723776	8.097408	14.814576	6.9564096
S.D.	0.29304911	0.17718316	0.4091957	1.63862072	0.26830671
Scenario-4					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	6.15080952	6.39219816	4.88420928	5.08219704	4.90230576
MIN	5.7602016	6.119064	4.6744128	4.784832	4.6192032
MAX	6.395112	6.7447728	5.267916	5.6911896	5.1252912
S.D.	0.12959266	0.13394726	0.13206007	0.19976559	0.14447176
Scenario-5					
	GA	PSO	HIWO	IWO	CTDHH
AVAERAGE	9.99079056	10.1150122	10.8631022	10.6689485	9.45096336
MIN	9.4224384	9.4960512	10.0113408	9.9193248	8.9807616
MAX	11.594016	11.1155328	13.5999648	13.4435376	10.535832
S.D.	0.5472875	0.40225822	0.99456773	0.6370966	0.35222608
Scenario-6					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	18.0885053	18.5375434	18.6669792	17.728416	17.6026608
MIN	16.838928	17.575056	17.2437984	16.9125408	16.7285088
MAX	22.08384	19.5441984	19.967472	19.1577312	18.8264736
S.D.	1.45430136	0.51147129	0.68799649	0.59463214	0.50735848
Scenario-7					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	8.69535864	8.48678904	8.37207576	8.074404	7.85341224
MIN	8.0836056	8.1802224	7.407288	7.70634	7.4855016
MAX	10.4576184	9.0083664	9.6570792	8.4792744	8.258436
S.D.	0.49896722	0.20209051	0.77423851	0.22832976	0.21165226
Scenario-8					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	16.0718213	15.7549795	15.2225136	15.410533	15.0498302
MIN	15.0814224	14.814576	14.5201248	14.6581488	14.5017216
MAX	17.529048	18.5320224	16.7193072	17.3082096	15.7439376
S.D.	0.48571107	0.91968838	0.50226504	0.64672523	0.32280923
Scenario-9					
	GA	PSO	HIWO	IWO	CTDHH
AVERAGE	34.1716752	35.8623158	35.3451859	36.5757466	31.5774374
MIN	32.1503904	32.2056	33.4570176	34.690032	29.8499904
MAX	38.0578176	49.5782208	38.2602528	39.7877184	34.3035648
S.D.	1.56576134	5.08242795	1.1431928	1.15834581	1.33380398

As discussed earlier in chapter 5 (simulation results), the total computational cost value of SWFS relies on the completion time (makespan) value and also depends on the VM available for each scenario. It can be concluded that the values of total computational cost

of larger datasets (scenarios seven, eight, and nine) is higher than those of the other smaller datasets (scenarios one to six). This mainly occurs due to the time required to execute the SWFA. The other important point is that the total computational cost of the proposed CTDHH approach always attains the lowest average value comparing with other baseline approaches. However, in scenario 4, the total computational cost value of HIWO approach is slightly lower than the total computational cost value of the proposed approach, and this due to the low convention time that HIWO appoach can achieve in most of the runs, which ultimately affect the produced optimal solution for each run.

Figure 6.8 presents five charts. Note that each chart is representing the relationship between the total computational cost ($/hour) and the considered scenarios for each of the approaches.

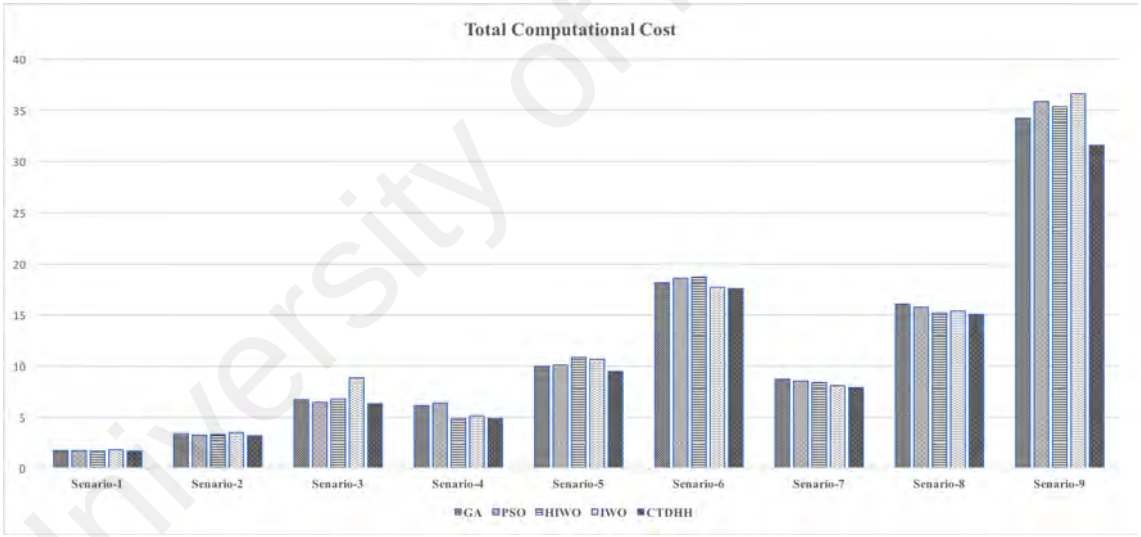

Figure 6.8: Average total computational cost of CTDHH and baselines for Montage

Figure 6.8 shows that the average cost values of scenarios three (comparing with scenarios one and two), scenario six (comparing with scenarios four and five) and scenario nine (comparing with scenarios seven and eight) are always higher than those of the other scenarios of all the approaches. This is due to the larger size of workflow tasks and more available VMs (8 VMs). Furthermore, the proposed CTDHH approach always attained

optimal performance results for all considered scenarios.

Figure 6.9 illustrates the average total computational cost ($/hour) values of all approaches for the considered scenarios.

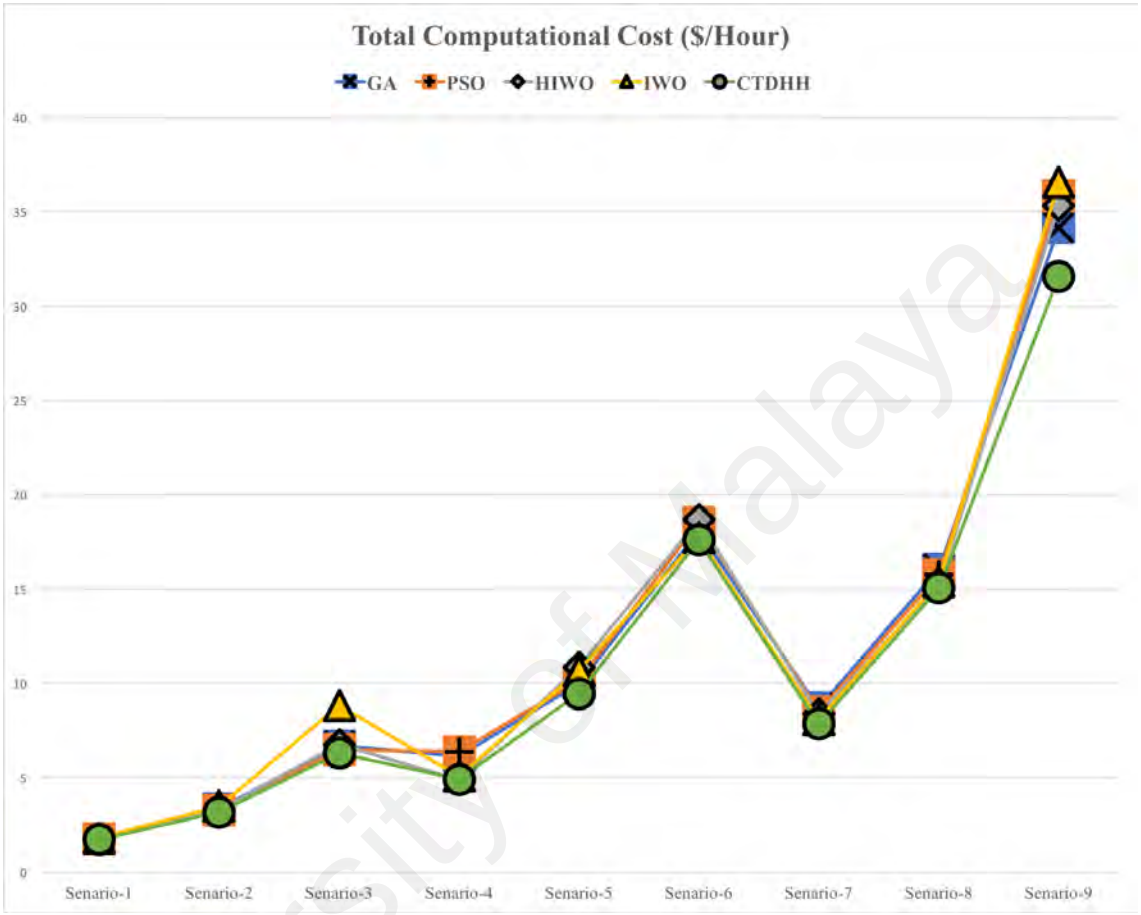

Figure 6.9: Average total computational of all approaches for Montage in real environment

Figure 6.9 shows as the size of dataset increases (with several available VMs), the proposed CTDHH approach performs better than other baseline approaches in terms of average total computational cost. Also, it can be evident that the average value of total computational cost of IWO approach achieves the worst value for most of the scenarios.

6.3.1.2 Comparison between Proposed CTDHH Approach and HHSA Approach

i. Completion time results

Table 6.4 illustrates the statistical results (average, minimum, maximum and S.D.) of completion time for Montage SWFA in the real-world environment for all considered scenarios.

Table 6.4: Completion time comparison between CTDHH and HHSA for Montage

Scenario-1		
	HHSA	CTDHH
AVERAGE	381.9	370.9333333
MIN	346	349
MAX	422	396
S.D.	14.81925588	14.64271817
Scenario-2		
	HHSA	CTDHH
AVAERAGE	359.9333333	344.6
MIN	334	326
MAX	380	382
S.D.	11.88720939	13.58142748
Scenario-3		
	HHSA	CTDHH
AVAERAGE	349.7	342.1666667
MIN	326	321
MAX	376	378
S.D.	10.51156998	14.57935085
Scenario-4		
	HHSA	CTDHH
AVAERAGE	1116.5	1065.533333
MIN	1055	1004
MAX	1200	1114
S.D.	30.08579686	31.40144224
Scenario-5		
	HHSA	CTDHH
AVAERAGE	1253.966667	1027.1
MIN	1179	976
MAX	1391	1145
S.D.	53.44315919	38.27878678
Scenario-6		
	HHSA	CTDHH
AVERAGE	1010.866667	956.5
MIN	961	909
MAX	1106	1023
S.D.	42.60705246	27.5690356
Scenario-7		
	HHSA	CTDHH
AVERAGE	1718.966667	1706.966667
MIN	1624	1627
MAX	1832	1795
S.D.	52.85013511	46.0033607
Scenario-8		
	HHSA	CTDHH
AVERAGE	1681.933333	1635.566667
MIN	1596	1576
MAX	1847	1711
S.D.	65.81370433	35.0818583
Scenario-9		
	HHSA	CTDHH
AVERAGE	1780.1	1715.866667
MIN	1645	1622
MAX	2245	1864
S.D.	110.0944266	72.47674221

The average completion time value of the proposed CTDHH approach is very close to

the average completion time of HHSA approach (Table 6.4). This is due to the utilized hyper-heuristic mechanism in employing the employed LLH algorithms (GA, PSO, IWO, HIWO) in order to find the most optimal SWFS solution.

Figure 6.10 presents two charts, where each of these charts is representing the relationship between the average completion time and the considered scenarios for each of the approaches.

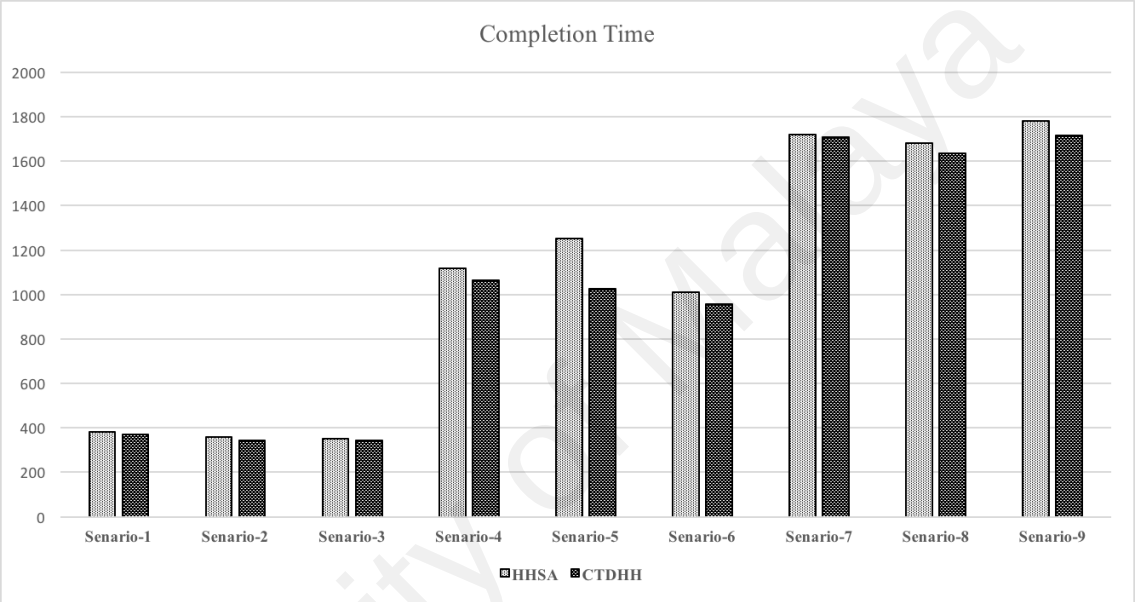

Figure 6.10: Average completion time of CTDHH and HHSA for Montage in real environment

It is evident from Figure 6.10 that the average completion time values of both approaches are significantly affected by the size of dataset (degree) as well as the availability of VMs.

The proposed CTDHH approach has the most optimal average completion time values. Ultimately, it concludes that the performance of the proposed CTDHH approach is improving with increased size of datasets compared to with HHSA approach.

Figure 6.11 shows the average completion time of all approaches for the considered scenarios.

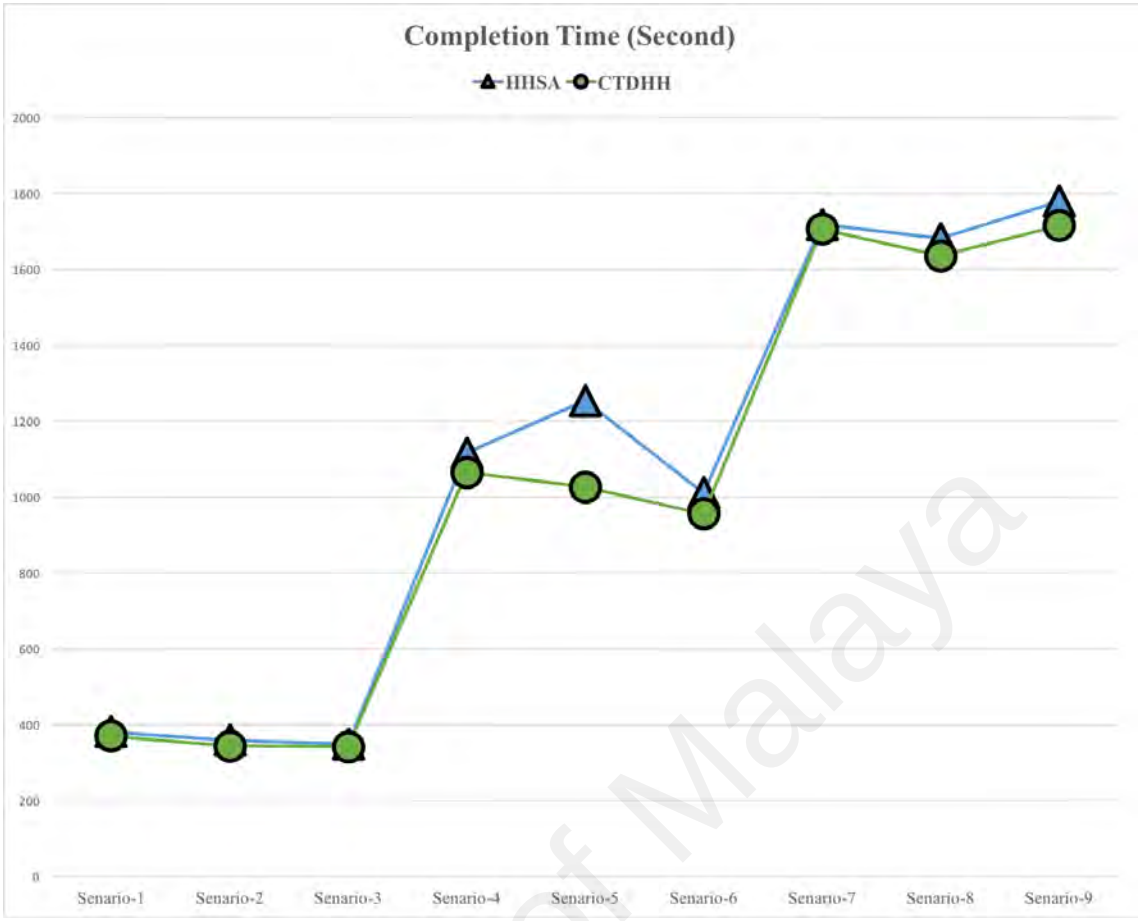

Figure 6.11: Average completion time of CTDHH and HHSA for Montage in real environment

It can be observed from Figure 6.11 that the proposed CTDHH approach are achieving a more optimal result when the problem size is getting bigger.

This is due to the hyper-heuristic method of selecting the most suitable LLH algorithm from the list of meta-heuristic algorithms based on the achieved time score instead of keep using the same algorithm for all runs.

ii. Total computational cost results

Table 6.5 presents the statistical results of total computational cost for Montage SWFA in the real-world environment for all considered scenarios.

Table 6.5: Average completion time of both approaches

Scenario-1		
	HHSA	CTDHH
AVERAGE	1.75704552	1.70659008
MIN	1.5918768	1.6056792
MAX	1.9415376	1.8219168
S.D.	0.06818043	0.06736822
Scenario-2		
	HHSA	CTDHH
AVAERAGE	3.31196256	3.17087136
MIN	3.0733344	2.9997216
MAX	3.496608	3.5150112
S.D.	0.10938135	0.12497086
Scenario-3		
	HHSA	CTDHH
AVAERAGE	6.43559904	6.2969616
MIN	5.9994432	5.9074272
MAX	6.9196032	6.9564096
S.D.	0.19344652	0.26830671
Scenario-4		
	HHSA	CTDHH
AVAERAGE	5.1367932	4.90230576
MIN	4.853844	4.6192032
MAX	5.52096	5.1252912
S.D.	0.13841873	0.14447176
Scenario-5		
	HHSA	CTDHH
AVAERAGE	11.5384997	9.45096336
MIN	10.8486864	8.9807616
MAX	12.7994256	10.535832
S.D.	0.49176257	0.35222608
Scenario-6		
	HHSA	CTDHH
AVERAGE	18.6031814	17.6026608
MIN	17.6854752	16.7285088
MAX	20.3539392	18.8264736
S.D.	0.78410611	0.50735848
Scenario-7		
	HHSA	CTDHH
AVERAGE	7.90862184	7.85341224
MIN	7.4716992	7.4855016
MAX	8.4286656	8.258436
S.D.	0.2431529	0.21165226
Scenario-8		
	HHSA	CTDHH
AVERAGE	15.4764778	15.0498302
MIN	14.6857536	14.5017216
MAX	16.9953552	15.7439376
S.D.	0.60559138	0.32280923
Scenario-9		
	HHSA	CTDHH
AVERAGE	32.7595363	31.5774374
MIN	30.273264	29.8499904
MAX	41.315184	34.3035648
S.D.	2.02608975	1.33380398

As previously discussed in chapter 5 (simulation results), the total computational cost value of SWFS heavily relies on the completion time (makespan) value and the VM available for each considered scenarios. The other important observation is that the total computational cost of the proposed CTDHH approach always attains the lowest average

value comparing with other HHSA approach.

Figure 6.12 presents two charts. Note that each chart is representing the relationship between the total computational cost ($/hour) and the considered scenarios for each of the approaches.

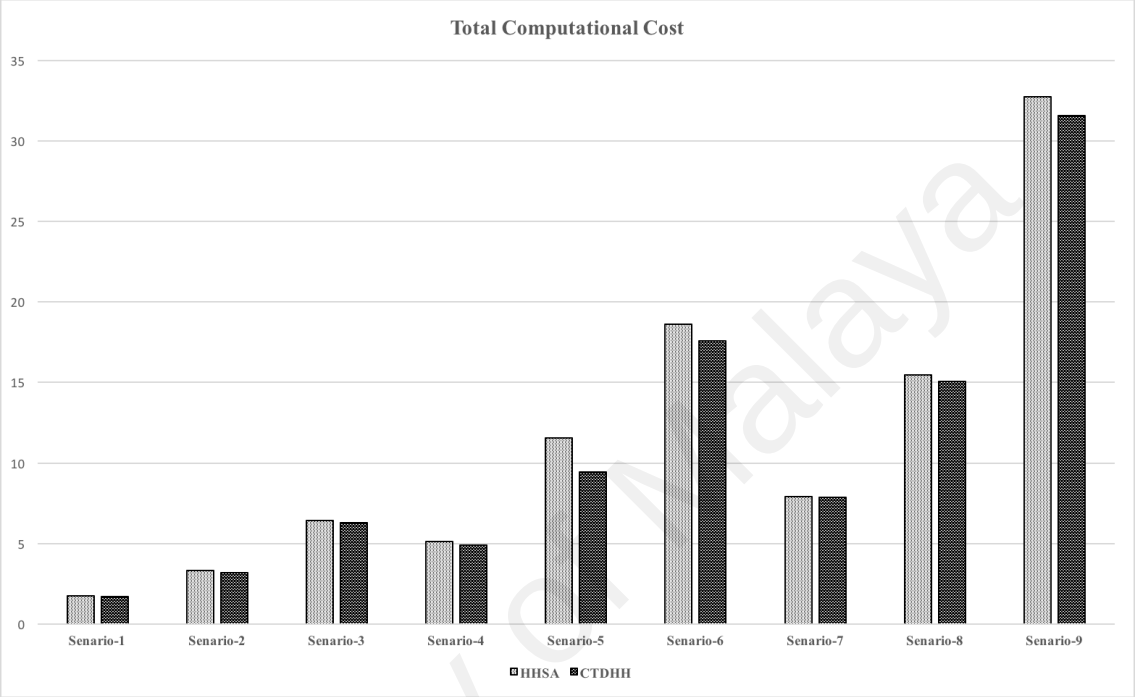

Figure 6.12: Average total computational cost of CTDHH and HHSA for Montage in real environment.

The proposed CTDHH approach has the optimal performance results for all considered scenarios (Figure 6.12). Furthermore, similar to the simulation results, the total computational cost value heavily depends on the completion time, number of available VMs as well as the number of workflow tasks for each of the considered scenarios.

Figure 6.13, illustrates the average total computational cost ($/hour) values of all approaches for the considered scenarios.

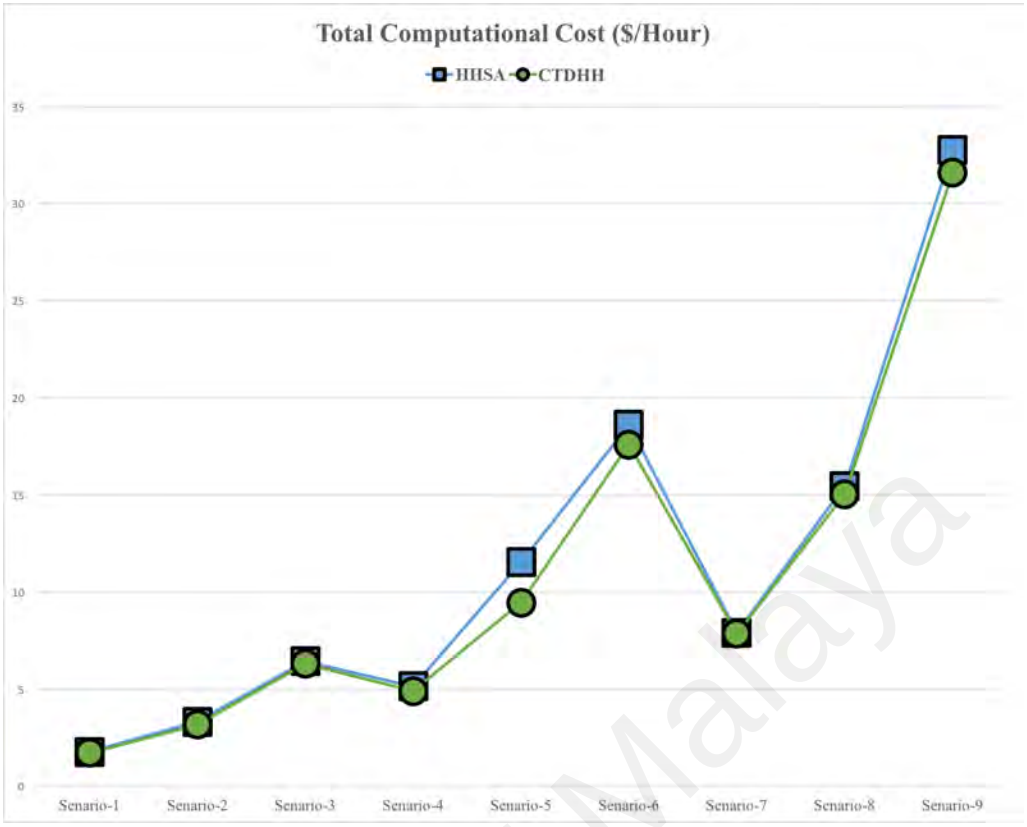

Figure 6.13: Average total computational of both approaches for Montage in real environment

From Figure 6.13, it can be concluded that in the real-world environment, the proposed approach is shows better performance that of compared with HHSA approach. This is due to the random behavior of original hyper-heuristic approach that can not get the most optimal results.

6.3.2 Normality Test and Significance Test of the Proposed Approach

As it has been discussed in Section 3.3.5, this section presents the statistical results of normality test and significance test of the proposed approach.

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
CTDHH	.113	30	.200*	.939	30	.088

Figure 6.14: Normality tests for the proposed CTDHH approach

Based on Figure 6.14, it can be concluded that result data is normally distributed, the paired T-test is used for statistical comparison, the result data from the proposed CTDHH approach with each of the baseline approaches as well as the comparison the result data from the proposed CTDHH approach and the HHSA.

The following Table 6.6 represents the T-Tests results by comparing proposed approach and the baseline approaches as well as the significance between the proposed CTDHH approach and the HHSA.

Table 6.6: Normality and significance tests for Montage

Scenario 1		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-2.449	0.0103105
CTDHH=PSO	-1.932	0.0316075
CTDHH=HIWO	-0.76	0.2267285
CTDHH=IWO	-5.249	0.0000065
CTDHH=HHSA	-2.7	0.0057225
Scenario 2		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-5.85	0.000001
CTDHH=PSO	-2.911	0.0034305
CTDHH=HIWO	-4.181	0.000122
CTDHH=IWO	-8.44	1.332E-09
CTDHH=HHSA	-4.72	0.0000275
Scenario 3		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-5.924	0.000001
CTDHH=PSO	-3.671	0.000485
CTDHH=HIWO	-6.664	1.31135E-07
CTDHH=IWO	-8.083	3.24495E-09
CTDHH=HHSA	-2.323	0.0137065
Scenario 4		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-30.043	1.0558E-23
CTDHH=PSO	-38.149	1.22275E-26
CTDHH=HIWO	0.562	0.2893465
CTDHH=IWO	-4.131	0.00014
CTDHH=HHSA	-5.601	0.0000025
Scenario 5		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-4.433	0.000061
CTDHH=PSO	-7.635	1.01745E-08
CTDHH=HIWO	-6.803	9.0325E-08
CTDHH=IWO	-8.652	7.902E-10
CTDHH=HHSA	-17.875	1.6837E-17
Scenario 6		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-1.79	0.0419715

CTDHH=PSO	-7.848	5.9005E-09
CTDHH=HIWO	-6.201	4.5936E-07
CTDHH=IWO	-1.001	0.162527
CTDHH=HHSA	-6.425	2.4985E-07
Scenario 7		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-8.379	1.5485E-09
CTDHH=PSO	-13.467	2.61525E-14
CTDHH=HIWO	-4.142	0.000136
CTDHH=IWO	-3.604	0.00058
CTDHH=HHSA	-1.183	0.123301
Scenario 8		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-10.428	1.2637E-11
CTDHH=PSO	-3.9	0.0002625
CTDHH=HIWO	-1.483	0.07448
CTDHH=IWO	-2.858	0.0039075
CTDHH=HHSA	-3.379	0.001047
Scenario 9		
	t-value	**Significat (1-tailed) test**
CTDHH=GA	-8.087	3.21955E-09
CTDHH=PSO	-4.954	0.0000145
CTDHH=HIWO	-10.825	5.302E-12
CTDHH=IWO	-15.923	3.56525E-16
CTDHH=HHSA	-3.169	0.001798

From Table 6.6 and based on the t-values and p-values, it can be observed that the proposed CTDHH approach has significant results for most of the Paired-Samples T-Test comparison results. This has ultimately confirmed that completion time and total computational cost results of proposed CTDHH are more optimised compared to the baseline and existing approaches for most the SWFA datasets. However, in the following exceptional cases, the proposed approach does not get significant results for the t-values and Paired-Samples T-Test (Table 6.6):

- In scenario 1 of Montage SWFA the proposed CTDHH approach does not get significant results comparing with HIWO approach.

- In scenario 4 of the same SWFA, the proposed CTDHH approach does not get significant results comparing with HIWO approach.

- In scenario 6 of the same SWFA, the proposed CTDHH approach does not get significant results comparing with IWO approach.

- In scenario 7 of the same SWFA, the proposed CTDHH approach does not get significant results comparing with HHSA approach.

The reason behind these exceptional results is the high complexity of the Montage SWFA cause significant differences between the achieved results of approaches.

6.4 Summary

This chapter discusses about the implementation and evaluation of the proposed CTDHH approach in the real-world environment. The behavior of the proposed CTDHH approach is more clear when it is implemented in the real cloud environment. It can be observed from results that the proposed CTDHH approach has the most optimal average completion time compared with the other baseline approaches. This is due to the fact that the HLH strategy of the proposed CTDHH approach depends on the TS of the execution results of employed LLH algorithms. Also, the average completion time values of all approaches are affected by the size of dataset (degree) as well as the availability of VMs. Thus, it can be also observed that in most cases, there are slight changes of the average value for scenarios one, two and three of all approaches and the performance of the proposed CTDHH approach is getting better as the size of dataset increases. This is because of the limited number of available VMs and the problem size is smaller comparing with other scenarios. And also this mainly occurs due to the time required to execute the SWFA. Furthermore, the average completion time value of the proposed CTDHH approach is very close to the average completion time of HHSA approach. This is due to the utilized hyper-heuristic mechanism in employing the employed LLH algorithms (GA, PSO, IWO, HIWO) in order to find the most optimal SWFS solution.

The total computational cost of the proposed CTDHH approach always attains the lowest average value comparing with other baseline approaches. Average total computational cost results show as the size of dataset increases (with several available VMs), the proposed

CTDHH approach performs better than other baseline approaches. This is due to the hyper-heuristic method of selecting the most suitable LLH algorithm from the list of meta-heuristic algorithms based on the achieved time score instead of keep using the same algorithm for all runs. The other important observation is that the total computational cost of the proposed CTDHH approach always attains the lowest average value comparing with other HHSA approach. This is due to the random behaviour of original hyper-heuristic approach that can not get the most optimal results.

It can also be concluded from normality test and significance test that result data is normally distributed, the paired T-test is used for statistical comparison, the result data from the proposed CTDHH approach with each of the baseline approaches as well as the comparison the result data from the proposed CTDHH approach and the HHSA. This has ultimately confirmed that completion time and total computational cost results of proposed CTDHH are more optimised compared to the baseline and existing approaches for most the SWFA datasets.

CHAPTER 7: CONCLUSION AND FUTURE WORK

This chapter concludes this research by presenting the following subsections: (i) summarising the key findings in relation to the research questions, (ii) listing the core contributions of the research, (iii) discussing the main limitations of the conducted research, and (iv) future work for the research.

7.1 Summary of Findings in Relation to the Research Questions

In Chapter 1, the formulated research objectives and Research Questions (RQs) have been formulated, which help in conducting the research methodology. In this section, the main finding of this research has been discussed by answering the formulated RQs.

RQ1: *What are the key cost optimisation challenges of SWFS in the cloud environment?*

Through an extensive literature review, the challenges of SWFS have been classified into three main types (Figure 2.6): (i) QoS performance; (ii) system functionality; and (ii) system architecture. Section 2.3.1, Section 2.3.2, and Section 2.3.3 provide complete details about these challenges.

RQ2: *What are the main cost optimisation aspects affecting the SWFS in the cloud environment?*

Several aspects need to be considered while scheduling the scientific workflow tasks. Section 2.1 presents a classification for aspects of cost optimisation SWFS approaches in a cloud computing environment based on eight main classes including computing environment, optimisation method, structural representation, profitability, scheduling technique, workload type, optimisation criteria, and QoS constraints.

RQ3: *What are the key cost parameters of SWFS in cloud computing, and how these parameters could affect the profitability of cost optimisation of SWFS?*

After analysing the cost optimisation parameters considered by different researchers

in the area of cost optimisation of SWFS in cloud computing, it has been found that the classification of the cost optimisation parameters is dependent on two types: (i) monetary cost parameters, and (ii) temporal cost parameters, as shown in Figure 2.3 in Section 2.2.

RQ4: *What are the existing cost optimisation approaches for the SWFS problem?*

The relevant cost optimisation approaches have been identified for SWFS problem in Section 2.4. It also provides a clearer understanding of the strengths of the underlying optimisation, and limitations for all considered and reviewed approaches. Furthermore, answering this question has helped in selecting the hyper-heuristic meta-heuristic approach as the most suitable proposed approach for the problem of this research.

RQ5: *How to model the cost optimisation problem of SWFS?*

In Section 4.1, the cost optimisation model of SWFS has been defined which helped to understand the mapping and scheduling processes of workflow tasks by considering the scheduling stages along with completion time and total computational cost parameters. Three standard stages of SWFS cost optimisation model are defined. The first stage discusses about the SWFAs, while the second stage describes the targeted computing environment. Finally, the third stage formulates the cost optimisation criteria.

RQ6: *How to propose a dynamic hyper-heuristic algorithm for cost optimisation challenge of SWFS?*

In Section 4.2, the Dynamic Hyper-Heuristic Algorithm (DHHA) has been presented for cost optimisation challenge of SWFS in a cloud environment. The DHHA is the main part of the proposed Completion Time Driven Hyper-Heuristic (CTDHH) approach. The proposed algorithm is considered as a new advanced technique that is capable of accelerating the run-time of a meta-heuristic algorithm. DHHA used the High Level Heuristic (HLH) strategy by employing four well-known population-based meta-heuristic algorithms, which act as the Low Level Heuristic (LLH). The main purpose of HLH

strategy is to intelligently guide the search process based on the performance of the employed meta-heuristic LLH algorithms.

RQ7: *What are the key experimental cloud environments that need to be considered for the evaluation of the proposed approach?*

From studying the existing experimental tools in the Section 3.3.1, it has been found that to comprehensively develop and evaluate the performance of any SWFS cost optimisation approach in cloud computing environment, there is a need to consider different computational environments. In this thesis, two different types of computational environments have been selected to evaluate the performance of the proposed CTDHH approach: (i) simulation-based using WorkflowSim, and (ii) a real-world based using Pegasus WfMS. The purpose of considering different computational environments is to efficiently understand the performance of the proposed CHDHH approach using diverse types of SWFAs.

RQ8: *What are the most relevant baseline and existing hyper-heuristic approaches that need to be considered to evaluate the proposed approach?*

In Chapter 5 and Chapter 6, the proposed CTDHH approach has been extensively evaluated. As explained earlier in Chapter 3 (Section 3.1.4), the performance of the proposed approach is evaluated by comparing it with four population-based approaches (i.e. GA, PSO, IWO, HIWO) and an existing hyper-heuristic approach named Hyper-Heuristic Scheduling Algorithm (HHSA) to effectively and efficiently evaluate the performance.

RQ9: *How to evaluate the computational-intensiveness and data-intensiveness of the proposed approach?*

In chapter 5 (Section 5.4), four types of SWFAs have been used to evaluate the performance of the proposed CTDHH approach. Note that these SWFAs also consider the computational-intensive of the workflow tasks that can be measured by the number and their composition structure. Furthermore, for each of these standard SWFA datasets,

there are different sizes that allows measuring the scalability of data-intensive applications (Section 5.4 and Section 6.3).

RQ10: *Would the proposed approach lead to results that are better than the considered baseline and existing hyper-heuristic approaches?*

From the several types of the conducted statistical analysis, the result of completion time parameter shows that the meta-heuristic algorithms (i.e. GA, PSO, IWO, and HWO) lack in attaining optimised results compared to the proposed CTDHH approach. This is due to the nature of solutions produce by these approaches (i.e. GA, PSO, IWO, and HWO), which are unable to provide an optimal solution for the SWFS, while the proposed CTDHH approach utilised a dynamic on-line learning strategy to find the most optimal solution. Furthermore, the results in Chapter 5 and Chapter 6 also show that the proposed CTDHH approach has achieved the most optimal results for most of the SWFA datasets and for most of the considered scenarios compared with the baseline and HHSA approaches.

For total computational cost results, similar to the completion time results, the proposed approach has achieved the cheapest total computational cost comparing with the baseline approaches. And these results are affected by the SWFA's type and size. This is mainly because of the complex and large size of the submitted workflow tasks, which ultimately make the SWFS approaches to take longer time to execute these tasks. Also, the Montage SWFA always consumes the lowest total cost compared with the other SWFA datasets (i.e., Epigenomics, Inspiral, and Sipht). This is due to the fact that the tasks of Montage SWFA have less precedence constraints compared to the other SWFA datasets.

7.2 Research Contributions

There are several contributions that have been gained from conducting this research. The following are the key contribution of this research:

- Several taxonomies of cost optimisation for SWFS challenges, aspects, and parameters.
- A completion time driven hyper-heuristic approach for cost optimisation of SWFS has been proposed. The proposed approach helps in optimising the completion time (makespan) and total execution cost of SWFS in the cloud computing environment.
- The proposed CTDHH approach can be profitable for service consumers, by reducing the total computational cost by utilising the computational resources of the cloud. At the same time, the proposed approach can provide more satisfying user requirements (i.e. shorter completion time and cheaper computational cost).
- The proposed approach helps in saving the energy and time of the service providers, by judiciously utilising the computational resources. This would ultimately help in reducing the computation cost as well as handling the computation-intensive and data-intensive SWFAs.
- This research would open new doors for a high impact research with innovative values through SWFA and cloud computing.
- To comprehensively evaluate the proposed approach, (i) two types of experimentation environment have been considered, (ii) several types of baseline and the existing hyper-heuristic approaches have been used to compare the proposed approach, (iii) several types of SWFAs have been considered, and (iv) different sizes of SWFAs have been used.
- Several papers have been published in high impact journals (please refer to the list of publications at the end of this thesis).

7.3 Research Limitations

The following are the key limitations of this research:

- In some cases, the proposed CTDHH approach has higher standard deviation values comparing with the other baseline and HHSA approaches.
- Due to the limited global contribution from remote research laboratory, only private cloud computing environment has been considered.
- In order to simplify the research problem, some cost parameters have to be assumed to be of constant value (e.g., cost of storage, communication cost).
- Only one type of SWFA has been considered in the real-world environment. This is mainly due to the long computational time, which has been taken to do the experimentation using a real scientific application with large data sizes. However, in the simulation based environment, four types of SWFAs have been considered.

7.4 Future Work

The following are the key future work that may be consider in the future study:

- Identify more classifications for the existing approaches in order to find a more optimal solution for the SWFS problem.
- Apply the proposed CTDHH approach in different kinds of optimisation challenges of SWFS including security, load balancing, reliability, and Quality of Service (QoS).
- Finding a more optimal solution by enhancing the selection and approval operators of the hyper-heuristic approaches.
- Implement the proposed approach in a hybrid cloud environment by linking the current private cloud environment to a public cloud environment by renting heterogeneous VMs from different service providers (e.g., Amazon). This would ultimately help in exploring the behavior of the proposed CTDHH approach with different computational environments, where the communication cost will also be considered.

REFERENCES

Abran, A., & Bourque, P. (2004). *Swebok: Guide to the software engineering body of knowledge* [Book]. IEEE Computer Society.

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

Afzal, A., Darlington, J., & McGough, A. (2006). Qos-constrained stochastic workflow scheduling in enterprise and scientific grids [Conference Proceedings]. In *Proceedings of the 7th ieee/acm international conference on grid computing* (p. 1-8). IEEE Computer Society.

Albodour, R., James, A., & Yaacob, N. (2012). High level qos-driven model for grid applications in a simulated environment [Journal Article]. *Future Generation Computer Systems*, *28*(7), 1133-1144.

Alkhanak, E. N., Lee, S. P., Rezaei, R., & Parizi, R. M. (2016). Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues. *Journal of Systems and Software*, *113*(Supplement C), 1 - 26. Retrieved from http://www.sciencedirect.com/science/article/pii/S0164121215002484

Amandeep Verma, S. K. (2012). Deadline and budget distribution based cost- time optimization workflow scheduling algorithm for cloud [Conference Proceedings]. In *International conference on recent advances and future trends in information technology (irafit2012)* (p. 1-4).

Amazon ec2 instance types. (2015). Amazon Web Services. Retrieved from https://aws.amazon.com/ec2/instance-types

Ambati, B. K., Ambati, J., & Mokhtar, M. M. (1991). Heuristic combinatorial optimization by simulated darwinian evolution: a polynomial time algorithm for the traveling salesman problem [Journal Article]. *Biological Cybernetics*, *65*(1), 31-35.

Anbazhagan Mani, A. N. (2002). *Understanding quality of service for web services* (Vol. 2014) [Web Page]. Retrieved from https://www.ibm.com/developerworks/library/ws-quality/

Arabnejad, H., & Barbosa, J. G. (2014). List scheduling algorithm for heterogeneous systems by an optimistic cost table [Journal Article]. *IEEE Transactions on Parallel and Distributed Systems*, *25*(3), 682-694.

Barrett, E., Howley, E., & Duggan, J. (2011). A learning architecture for scheduling workflow applications in the cloud [Conference Proceedings]. In *2011 ninth ieee european conference on web services (ecows)* (p. 83-90). IEEE.

Berman, F., Casanova, H., Chien, A., & Cooper, K., Kennedy. (2005). New grid scheduling and rescheduling methods in the grads project [Journal Article]. *International Journal of Parallel Programming*, *33*(2-3), 209-229.

Bhise, V. K., & Mali, A. S. (2013). Ec2 instance provisioning for cost optimization [Conference Proceedings]. In *2013 international conference on advances in computing, communications and informatics (icacci)* (p. 1891-1895). IEEE.

Bittencourt, L. F., Madeira, E. R., & Da Fonseca, N. L. (2012). Scheduling in hybrid clouds [Journal Article]. *IEEE Communications Magazine*, *50*(9), 42-47.

Bittencourt, L. F., & Madeira, E. R. M. (2011). Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds [Journal Article]. *Journal of Internet Services and Applications*, *2*(3), 207-227.

Bittencourt, L. F., & Madeira, E. R. M. (2013). Using time discretization to schedule scientific workflows in multiple cloud providers [Conference Proceedings]. In *2013 ieee sixth international conference on cloud computing (cloud)* (p. 123-130). IEEE.

Bittencourt, L. F., Senna, C. R., & Madeira, E. R. (2010). Scheduling service workflows for cost optimization in hybrid clouds [Conference Proceedings]. In *2010 international conference on network and service management (cnsm)* (p. 394-397). IEEE.

Bjorkqvist, M., Chen, L. Y., & Binder, W. (2012). Cost-driven service provisioning in hybrid clouds [Conference Proceedings]. In *2012 5th ieee international conference on service-oriented computing and applications (soca)* (p. 1-8). IEEE.

Bourque, P., & Fairley, R. E. (2014). *Guide to the software engineering body of knowledge (swebok (r)): Version 3.0* [Book]. IEEE Computer Society Press.

Broberg, J., Buyya, R., & Tari, Z. (2009). Metacdn: Harnessing 'storage clouds' for high performance content delivery [Journal Article]. *Journal of Network and Computer Applications*, *32*(5), 1012-1022.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (2009). Exploring hyper-heuristic methodologies with genetic programming. In *Computational intelligence* (pp. 177–201). Springer.

Burke, E. K., Kendall, G., et al. (2005). *Search methodologies*. Springer.

Buyya, R., & Murshed, M. (2002). Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and computation: practice and experience*, *14*(13-15), 1175–1220.

Byun, E.-K., Kee, Y.-S., Kim, J.-S., & Maeng, S. (2011). Cost optimized provisioning of elastic resources for application workflows [Journal Article]. *Future Generation Computer Systems*, *27*(8), 1011-1026.

Calheiros, R. N., Netto, M. A., De Rose, C. A., & Buyya, R. (2013). Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications [Journal Article]. *Software: Practice and Experience*, *43*(5), 595-612.

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

Chandrakumar, B. (2013). Time synchronization on cognitive radio ad hoc networks: A bio-inspired approach [Journal Article]. *Computing*, 115.

Chen, T., Bahsoon, R., & Theodoropoulos, G. (2013). Dynamic qos optimization architecture for cloud-based dddas [Journal Article]. *Procedia Computer Science*, *18*, 1881-1890.

Chen, W., Da Silva, R. F., Deelman, E., & Sakellariou, R. (2013). Balanced task clustering in scientific workflows [Conference Proceedings]. In *escience (escience), 2013 ieee 9th international conference on* (p. 188-195). IEEE.

Chen, W., da Silva, R. F., Deelman, E., & Sakellariou, R. (2015). Using imbalance metrics to optimize task clustering in scientific workflow executions [Journal Article]. *Future Generation Computer Systems*, *46*, 69-84.

Chen, W., & Deelman, E. (n.d.). Workflowsim: A toolkit for simulating scientific workflows in distributed environments [Conference Proceedings]. In *E-science (e-science), 2012 ieee 8th international conference on* (p. 1-8). IEEE.

Chen, W., Ferreira da Silva, R., Deelman, E., & Fahringer, T. (2015). Dynamic and fault-tolerant clustering for scientific workflows [Journal Article].

Chen, W.-n., Shi, Y., & Zhang, J. (2009). An ant colony optimization algorithm for the time-varying workflow scheduling problem in grids [Conference Proceedings]. In *Cec'09. ieee congress on evolutionary computation, 2009* (p. 875-880). IEEE.

Chen, W.-N., & Zhang, J. (2009). An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements [Journal Article]. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *39*(1), 29-43.

Choudhary, V., Kacker, S., Choudhury, T., & Vashisht, V. (2012). An approach to improve task scheduling in a decentralized cloud computing environment [Journal Article]. *International Journal of Computer Technology and Applications*, *3*(1), 312-316.

Chunlin, L., & Layuan, L. (2006). Qos based resource scheduling by computational economy in computational grid [Journal Article]. *Information Processing Letters*, *98*(3), 119-126.

Coalition, W. M. (2005). [Standard]. Retrieved from `http://www.wfmc.org/resources`

Cowling, P., Kendall, G., & Soubeiga, E. (2000). A hyperheuristic approach to scheduling a sales summit. In *International conference on the practice and theory of automated timetabling* (pp. 176–190).

Cowling, P. I., & Chakhlevitch, K. (2007). Using a large set of low level heuristics in a hyperheuristic approach to personnel scheduling [Book Section]. In *Evolutionary scheduling* (p. 543-576). Springer.

Czarnul, P. (2013). Modeling, run-time optimization and execution of distributed workflow

applications in the jee-based beesycluster environment [Journal Article]. *The Journal of Supercomputing*, *63*(1), 46-71.

Deelman, E., Blythe, J., Gil, Y., & Kesselman, M., Livny. (n.d.). Pegasus: Mapping scientific workflows onto the grid [Conference Proceedings]. In *Grid computing* (p. 11-20). Springer.

Deelman, E., Juve, G., Rynge, M., Voeckler, J., & Berriman, G. (2013). Comparing futuregrid, amazon ec2, and open science grid for scientific workflows [Journal Article]. *Computing in Science and Engineering*, *15*(4), 20-29.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, G. B., & Good, J. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems [Journal Article]. *Scientific Programming*, *13*(3), 219-237.

Deelman, E., Vahi, K., Juve, G., Rynge, R. F., & Livny, M. (2015). Pegasus, a workflow management system for science automation [Journal Article]. *Future Generation Computer Systems*, *46*, 17-35.

Delavar, A. G., & Aryan, Y. (2012). A goal-oriented workflow scheduling in heterogeneous distributed systems [Journal Article]. *International Journal of Computer Applications*, *52*(8), 27-33.

Deldari, A., Naghibzadeh, M., & Abrishami, S. (2017). Cca: a deadline-constrained workflow scheduling algorithm for multicore resources on the cloud. *The journal of Supercomputing*, *73*(2), 756–781.

Deng, K., Kong, L., Song, J., Ren, K., & Yuan, D. (2011). A weighted k-means clustering based co-scheduling strategy towards efficient execution of scientific workflows in collaborative cloud environments [Conference Proceedings]. In *2011 ieee ninth international conference on dependable, autonomic and secure computing (dasc)* (p. 547-554). IEEE.

de Oliveira, D., Ocaña, K. A., Baião, F., & Mattoso, M. (2012). A provenance-based adaptive scheduling heuristic for parallel scientific workflows in clouds [Journal Article]. *Journal of Grid Computing*, *10*(3), 521-552.

de Oliveira, D., Viana, V., Ogasawara, E., Ocaña, K., & Mattoso, M. (2013). Dimensioning the virtual cluster for parallel scientific workflows in clouds [Conference Proceedings]. In *Proceedings of the 4th acm workshop on scientific cloud computing*

(p. 5-12). ACM.

Dong, F. (2009). *Workflow scheduling algorithms in the grid* (Thesis). Queen's University, Kingston, Ontario, Canada.

Duan, R., Prodan, R., & Li, X. (2012). Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds [Journal Article]. *Provider*.

Durillo, J. J., Nae, V., & Prodan, R. (n.d.). Multi-objective workflow scheduling: An analysis of the energy efficiency and makespan tradeoff [Conference Proceedings]. In *2013 13th ieee/acm international symposium on cluster, cloud and grid computing (ccgrid)* (p. 203-210). IEEE.

Durillo, J. J., Prodan, R., & Fard, H. M. (2012). Moheft: a multi-objective list-based method for workflow scheduling [Conference Proceedings]. In *Proceedings of the 2012 ieee 4th international conference on cloud computing technology and science (cloudcom)* (p. 185-192). IEEE Computer Society.

Dutta, K., & VanderMeer, D. (2011). Cost-based decision-making in middleware virtualization environments [Journal Article]. *European Journal of Operational Research*, *210*(2), 344-357.

Fang, H.-L., Ross, P., & Corne, D. (1993). *A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems* [Book]. University of Edinburgh, Department of Artificial Intelligence.

Fida, A. (2008). *Workflow scheduling for service oriented cloud computing* (Thesis). University of Saskatchewan.

Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud computing and grid computing 360-degree compared [Conference Proceedings]. In *Gce'08 grid computing environments workshop* (p. 1-10). IEEE.

Gao, Y., Ma, H., Zhang, H., Kong, X., & Wei, W. (2013). Concurrency optimized task scheduling for workflows in cloud [Conference Proceedings]. In *2013 ieee sixth international conference on cloud computing (cloud)* (p. 709-716). IEEE.

Garg, S. K., Buyya, R., & Siegel, H. J. (2010). Time and cost trade-off management for scheduling parallel applications on utility grids [Journal Article]. *Future*

Generation Computer Systems, *26*(8), 1344-1355.

Genez, T. A., Bittencourt, L. F., & Madeira, E. R. (2012). Workflow scheduling for saas/paas cloud providers considering two sla levels [Conference Proceedings]. In *2012 ieee network operations and management symposium (noms)* (p. 906-912). IEEE.

Glover, F. W., & Kochenberger, G. A. (2006). *Handbook of metaheuristics* (Vol. 57). Springer Science & Business Media.

Grandinetti, L., Pisacane, O., & Sheikhalishahi, M. (2013). An approximate constraint method for a multi-objective job scheduling in the cloud [Journal Article]. *Future Generation Computer Systems*, *29*(8), 1901–1908.

Grossman, R. L., Gu, Y., Sabala, M., & Zhang, W. (2009). Compute and storage clouds using wide area high performance networks [Journal Article]. *Future Generation Computer Systems*, *25*(2), 179-183.

Grounds, N., Antonio, J., & Muehring, J. (2009). Cost-minimizing scheduling of workflows on a cloud of memory managed multicore machines [Book Section]. In *Cloud computing* (Vol. 5931, p. 435-450). Springer Berlin Heidelberg.

Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., . . . Vishnu, A. (2014). A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems [Journal Article]. *Computing*, 1-24.

Harman, M., Lakhotia, K., Singer, J., White, D. R., & Yoo, S. (2013). Cloud engineering is search based software engineering too [Journal Article]. *Journal of Systems and Software*, *86*(9), 2225-2241.

Hirales-Carbajal, A., Tchernykh, A., Yahyapour, R., González-García, J. L., Röblitz, T., & Ramírez-Alcaraz, J. M. (2012). Multiple workflow scheduling strategies with user run time estimates on a grid [Journal Article]. *Journal of Grid Computing*, *10*(2), 325-346.

Horiuchi, M., & Taura, K. (2012). Acceleration of data-intensive workflow applications by using file access history [Conference Proceedings]. In *2012 sc companion: High performance computing, networking, storage and analysis (scc)* (p. 157-165). IEEE.

Hsu, C.-H., Cuzzocrea, A., & Chen, S.-C. (2011). Cad: An efficient data management and migration scheme across clouds for data-intensive scientific applications [Book Section]. In *Data management in grid and peer-to-peer systems* (Vol. 6864, p. 120-134). Springer Berlin Heidelberg.

Hussin, N. (2005). Tabu search based hyper-heuristic approaches for examination timetabling. *School of Computer Science and Information Technology. University of Nottingham*.

Islam, M., Huang, A. K., Battisha, A., & Abdelnur, A. (n.d.). Oozie: towards a scalable workflow management system for hadoop [Conference Proceedings]. In *Proceedings of the 1st acm sigmod workshop on scalable workflow execution engines and technologies* (p. 4). ACM.

ITU. Recommendation, E. (n.d.). *Quality of service and dependability vocabulary.* [Standard].

Jiang, H.-J., Huang, K.-C., Chang, H.-Y., Gu, D.-S., & Shih, P.-J. (2011). Scheduling concurrent workflows in hpc cloud through exploiting schedule gaps [Book Section]. In (p. 282-293). Springer.

Jing, H., Kai, W., Lok Kei, L., Seungbeom, M., , & Melody, M. (2013). A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing [Journal Article]. *International Journal of Soft Computing and Software Engineering [JSCSE]*, *3*(3), 351-358.

Jrad, F., Tao, J., Brandic, I., & Streit, A. (2014). Multi-dimensional resource allocation for data-intensive large-scale cloud applications [Conference Proceedings]. In *Closer* (p. 691-702).

Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., & Vahi, K. (2013). Characterizing and profiling scientific workflows [Journal Article]. *Future Generation Computer Systems*, *29*(3), 682-692.

Kaur, N., Aulakh, T., & Cheema, R. (2011). Comparison of workflow scheduling algorithms in cloud computing [Journal Article]. *International Journal of Advanced Computer Science and Applications*, *2*(10), 81-86.

Kaur, S., & Verma, A. (2012). An efficient approach to genetic algorithm for task scheduling in cloud computing environment [Journal Article]. *International*

Journal of Information Technology and Computer Science (IJITCS), *4*(10), 74-79.

Khajemohammadi, H., Fanian, A., & Gulliver, T. A. (2013). Fast workflow scheduling for grid computing based on a multi-objective genetic algorithm [Conference Proceedings]. In *2013 ieee pacific rim conference on communications, computers and signal processing (pacrim)* (p. 96-101). IEEE.

Koseoglu, M., & Karasan, E. (2010). Joint resource and network scheduling with adaptive offset determination for optical burst switched grids [Journal Article]. *Future Generation Computer Systems*, *26*(4), 576-589.

Kousalya, G., Balakrishnan, P., & Raj, C. P. (2017). Workflow scheduling algorithms and approaches. In *Automated workflow scheduling in self-adaptive clouds* (pp. 65–83). Springer.

Kwok, Y.-K., & Ahmad, I. (1999). Static scheduling algorithms for allocating directed task graphs to multiprocessors [Journal Article]. *ACM Computing Surveys (CSUR)*, *31*(4), 406-471.

Li, J., Su, S., Cheng, X., Huang, Q., & Zhang, Z. (2011). Cost-conscious scheduling for large graph processing in the cloud [Conference Proceedings]. In *2011 ieee 13th international conference on high performance computing and communications (hpcc)* (p. 808-813). IEEE.

Li, J., Su, S., Cheng, X., Song, M., Ma, L., & Wang, J. (2015). Cost-efficient coordinated scheduling for leasing cloud resources on hybrid workloads [Journal Article]. *Parallel Computing*, *44*(0), 1-17. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0167819115000332`

Li, K., Li, S., Xu, Y., & Xie, Z. (n.d.). A dag task scheduling scheme on heterogeneous computing systems using invasive weed optimization algorithm [Conference Proceedings]. In *2014 sixth international symposium on parallel architectures, algorithms and programming* (p. 262-267). IEEE.

Lin, C., & Lu, S. (2011). Scheduling scientific workflows elastically for cloud computing [Conference Proceedings]. In *2011 ieee international conference on cloud computing (cloud)* (p. 746-747). IEEE.

Lin, J., Luo, D., Li, X., Gao, K., & Liu, Y. (2017). Differential evolution based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time.

In *Asia-pacific conference on simulated evolution and learning* (pp. 75–86).

Lin, X., & Wu, C. Q. (2013). On scientific workflow scheduling in clouds under budget constraint [Conference Proceedings]. In *2013 42nd international conference on parallel processing (icpp)* (p. 90-99). IEEE.

Lingfang, Z., Veeravalli, B., & Xiaorong, L. (2012). Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud [Conference Proceedings]. In *2012 ieee 26th international conference on advanced information networking and applications (aina)* (p. 534-541).

Liu, B., Wu, Y., Xu, X., Hu, N., & Cheng, X. (2014). Chaos adaptive improved particle swarm algorithm for solving multi-objective optimization [Journal Article]. *Telkomnika Indonesian Journal of Electrical Engineering*, *12*(1), 703-710.

Liu, H., Xu, D., & Miao, H. (2011). Ant colony optimization based service flow scheduling with various qos requirements in cloud computing [Conference Proceedings]. In *2011 first acis international symposium on software and network engineering (ssne)* (p. 53-58). IEEE.

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

Liu, L., Zhang, M., Buyya, R., & Fan, Q. (2017). Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing. *Concurrency and Computation: Practice and Experience*, *29*(5).

Liu, X. (2011). *A novel probabilistic temporal framework and its strategies for cost-effective delivery of high qos in scientific cloud workflow systems* (Thesis). Faculty of Information and Communication Technologies.

Liu, X. (2012). *The design of cloud workflow systems* [Book]. Springer.

Liu, X., Chen, J., Wu, Z., Ni, Z., Yuan, D., & Yang, Y. (2010). Handling recoverable temporal violations in scientific workflow systems: a workflow rescheduling based strategy [Conference Proceedings]. In *Proceedings of the 2010 10th ieee/acm international conference on cluster, cloud and grid computing* (p. 534-537). IEEE Computer Society.

Liu, X., Ni, Z., Wu, Z., Yuan, D., Chen, J., & Yang, Y. (2011). A novel general framework for automatic and cost-effective handling of recoverable temporal violations in scientific workflow systems [Journal Article]. *Journal of Systems and Software*, *84*(3), 492-509.

Liu, Y., Zhu, X., Chen, Y., Li, Y., & Deng, N. (2014). Automatic building process of self-closed modified n-tree [Journal Article]. *Telkomnika Indonesian Journal of Electrical Engineering*, *12*(1), 157-166.

Lombardi, F., & Di Pietro, R. (2011). Secure virtualization for cloud computing [Journal Article]. *Journal of Network and Computer Applications*, *34*(4), 1113-1122.

Ludscher, B., Altintas, I., Berkley, C., Higgins, J., & Zhao, Y. (2006). Scientific workflow management and the kepler system [Journal Article]. *Concurrency and Computation: Practice and Experience*, *18*(10), 1039-1065.

Ma, Y., Gong, B., & Zou, L. (2009). Marginal pricing based scheduling strategy of scientific workflow using cost-gradient metric [Conference Proceedings]. In *Gcc'09. eighth international conference on grid and cooperative computing, 2009* (p. 136-143). IEEE.

Malawski, M., Figiela, K., Bubak, M., Deelman, E., & Nabrzyski, J. (2014). Cost optimization of execution of multi-level deadline-constrained scientific workflows on clouds [Book Section]. In (p. 251-260). Springer.

Malawski, M., Juve, G., Deelman, E., & Nabrzyski, J. (2012). Cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds [Conference Proceedings]. In *Proceedings of the international conference on high performance computing, networking, storage and analysis* (p. 22). IEEE Computer Society Press.

Malik, S., Nazir, B., Qureshi, K., & Khan, I. A. (2013). A reliable checkpoint storage strategy for grid [Journal Article]. *Computing*, *95*(7), 611-632.

Manvi, S. S., & Krishna Shyam, G. (2013). Resource management for infrastructure as a service (iaas) in cloud computing: A survey [Journal Article]. *Journal of Network and Computer Applications*, *41*, 424–440.

Mao, L., Yang, Y., & Xu, H. (2013). Design and optimization of cloud-oriented workflow system [Journal Article]. *Journal of Software*, *8*(1), 251-258.

Mao, M., & Humphrey, M. (2011). Auto-scaling to minimize cost and meet application deadlines in cloud workflows [Conference Proceedings]. In *2011 international conference for high performance computing, networking, storage and analysis (sc)* (p. 1-12). IEEE.

Mell, P., & Grance, T. (2009). The nist definition of cloud computing [Journal Article]. *National Institute of Standards and Technology*, *53*(6), 50.

Menasce, D. A., & Casalicchio, E. (2004). A framework for resource allocation in grid computing [Conference Proceedings]. In *The ieee computer society's 12th annual international symposium on modeling, analysis, and simulation of computer and telecommunications systems, 2004. (mascots 2004).* (p. 259-267).

Miu, T., & Missier, P. (2012). Predicting the execution time of workflow activities based on their input features [Conference Proceedings]. In *2012 sc companion: High performance computing, networking, storage and analysis (scc)* (p. 64-72). IEEE.

Nareyek, A. (2003). Choosing search heuristics by non-stationary reinforcement learning [Book Section]. In *Metaheuristics: Computer decision-making* (p. 523-544). Springer.

Nargunam, K. L. G., & Shajin, A. (2012). Compatibility of hybrid process scheduler in green it cloud computing environment [Journal Article]. *International Journal of Computer Applications*, *55*(5), 27-33.

Netjinda, N., Sirinaovakul, B., & Achalakul, T. (2012). Cost optimization in cloud provisioning using particle swarm optimization [Conference Proceedings]. In *2012 9th international conference on electrical engineering/electronics, computer, telecommunications and information technology (ecti-con)* (p. 1-4). IEEE.

Ostermann, S., Prodan, R., & Fahringer, T. (2010). Dynamic cloud provisioning for scientific grid workflows [Conference Proceedings]. In *2010 11th ieee/acm international conference on grid computing (grid)* (p. 97-104). IEEE.

Pacini, E., Mateos, C., & Garino, C. G. (2014). Multi-objective swarm intelligence schedulers for online scientific clouds [Journal Article]. *Computing*, 1-28.

Pandey, S., Wu, L., Guru, S. M., & Buyya, R. (2010). A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments [Conference Proceedings]. In *2010 24th ieee international conference on advanced*

information networking and applications (aina) (p. 400-407). IEEE.

Park, J., Mei, Y., Nguyen, S., Chen, G., & Zhang, M. (2017). An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. *Applied Soft Computing*.

Pegasus workflow repository-workflow generator. (2015). Web Page. Retrieved from `https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator`

Poola, D., Garg, S. K., Buyya, R., Yang, Y., & Ramamohanarao, K. (2014). Robust scheduling of scientific workflows with deadline and budget constraints in clouds [Conference Proceedings]. In *2014 ieee 28th international conference on advanced information networking and applications (aina)* (p. 858-865). IEEE.

Prodan, R., & Wieczorek, M. (2010). Bi-criteria scheduling of scientific grid workflows [Journal Article]. *IEEE Transactions on Automation Science and Engineering*, *7*(2), 364-376.

Rahman, M., Li, X., & Palit, H. (2011). Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment [Conference Proceedings]. In *2011 ieee international symposium on parallel and distributed processing workshops and phd forum (ipdpsw)* (p. 966-974). IEEE.

Ramakrishnan, L., Chase, J. S., Gannon, D., Nurmi, D., & Wolski, R. (2011). Deadline-sensitive workflow orchestration without explicit resource control [Journal Article]. *Journal of Parallel and Distributed Computing*, *71*(3), 343-353.

Ranaldo, N., & Zimeo, E. (2009). Time and cost-driven scheduling of data parallel tasks in grid workflows [Journal Article]. *IEEE Systems Journal*, *3*(1), 104-120.

Rezaei, R., Chiew, T. K., Lee, S. P., & Shams Aliee, Z. (2014). Interoperability evaluation models: A systematic review [Journal Article]. *Computers in Industry*, *65*(1), 1-23.

Rodriguez, M. A., & Buyya, R. (2017). Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms. *Future Generation Computer Systems*.

Saeid Abrishami, D. E., Mahmoud Naghibzadeh. (2012). Cost-driven scheduling of grid

workflows using partial critical paths [Journal Article]. *IEEE Transactions on Parallel and Distributed Systems*, *23*(8), 1400-1414.

Saeid Abrishami, D. E., Mahmoud Naghibzadeh. (2013). Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds [Journal Article]. *Future Generation Computer Systems*, *29*(1), 158-169.

Sahar Adabi, A. M. R., Rahmani Ali. (2014). Bi-level fuzzy based advanced reservation of cloud workflow applications on distributed grid resources [Journal Article]. *The Journal of Supercomputing*, *67*(1), 175-218.

Sakellariou, R., & Zhao, H. (2004a). A hybrid heuristic for dag scheduling on heterogeneous systems [Conference Proceedings]. In *Proceedings. 18th international parallel and distributed processing symposium, 2004.* (p. 111). IEEE.

Sakellariou, R., & Zhao, H. (2004b). A low-cost rescheduling policy for efficient mapping of workflows on grid systems [Journal Article]. *Scientific Programming*, *12*(4), 253-262.

Sakellariou, R., Zhao, H., Tsiakkouri, E., & Dikaiakos, M. D. (2007). Scheduling workflows with budget constraints [Book Section]. In *Integrated research in grid computing* (p. 189-202). Springer.

Salehi, M. A., & Buyya, R. (2010). Adapting market-oriented scheduling policies for cloud computing [Book Section]. In *Algorithms and architectures for parallel processing* (Vol. 6081, p. 351-362). Springer.

Sang, H.-y., & Pan, Q.-k. (n.d.). An effective invasive weed optimization algorithm for the flow shop scheduling with intermediate buffers [Conference Proceedings]. In *2013 25th chinese control and decision conference (ccdc)* (p. 861-864). IEEE.

Senna, C. R., Bittencourt, L. F., & Madeira, E. R. (2012). Service workflow monitoring in private clouds: The user point of view [Conference Proceedings]. In *2012 ieee latin america conference on cloud computing and communications (latincloud)* (p. 25-30). IEEE.

Sharif, S., Taheri, J., Zomaya, A. Y., & Nepal, S. (2013). Mphc: Preserving privacy for workflow execution in hybrid clouds [Conference Proceedings]. In *Proceedings of the 2013 international conference on parallel and distributed computing, applications and technologies* (p. 272-280). IEEE Computer Society.

Sharma, R., Nayak, N., Krishnanand, K., & Rout, P. (n.d.). Modified invasive weed optimization with dual mutation technique for dynamic economic dispatch [Conference Proceedings]. In *Energy, automation, and signal (iceas), 2011 international conference on* (p. 1-6). IEEE.

Shi, J. (2014). A budget and deadline aware scientific workflow resource provisioning and scheduling mechanism for cloud [Conference Proceedings]. In *Proceedings of the 2014 ieee 18th international conference on computer supported cooperative work in design (cscwd)* (p. 672-677). IEEE.

Srinivasan, P. (2012). Time-cost scheduling algorithm [Conference Proceedings]. In *International conference on computing and control engineering (iccce 2012)* (p. 1-5).

Stephanakis, I. M., Chochliouros, I. P., Caridakis, G., & Kollias, S. (2013). A particle swarm optimization (pso) model for scheduling nonlinear multimedia services in multicommodity fat-tree cloud networks [Book Section]. In *Engineering applications of neural networks* (Vol. 384, p. 257-268). Springer.

Stevens, T., De Leenheer, M., Develder, C., Dhoedt, B., Christodoulopoulos, K., Kokkinos, P., & Varvarigos, E. (2009). Multi-cost job routing and scheduling in grid networks [Journal Article]. *Future Generation Computer Systems*, *25*(8), 912-925.

Storer, R. H., Wu, S. D., & Vaccari, R. (1992). New search spaces for sequencing problems with application to job shop scheduling [Journal Article]. *Management science*, *38*(10), 1495-1509.

Sudha, M., & Monica, M. (2012). Dynamic adaptive workflow scheduling for instance intensive cloud applications [Journal Article]. *Journal of Expert Systems*, *1*(1), 31-36.

Szabo, C., & Kroeger, T. (2012). Evolving multi-objective strategies for task allocation of scientific workflows on public clouds [Conference Proceedings]. In *2012 ieee congress on evolutionary computation (cec)* (p. 1-8). IEEE.

Szabo, C., Sheng, Q. Z., Kroeger, T., Zhang, Y., & Yu, J. (2014). Science in the cloud: Allocation and execution of data-intensive scientific workflows [Journal Article]. *Journal of Grid Computing*, *12*(2), 245-264.

Talukder, A., Kirley, M., & Buyya, R. (2007). Multiobjective differential evolution

for workflow execution on grids [Conference Proceedings]. In *Proceedings of the 5th international workshop on middleware for grid computing: held at the acm/ifip/usenix 8th international middleware conference* (p. 3). ACM.

Talukder, A., Kirley, M., & Buyya, R. (2009). Multiobjective differential evolution for scheduling workflow applications on global grids [Journal Article]. *Concurrency and Computation: Practice and Experience*, *21*(13), 1742-1756.

Tan, W., Sun, Y., Li, L. X., Lu, G., & Wang, T. (2013). A trust service-oriented scheduling model for workflow applications in cloud computing [Journal Article]. *IEEE Systems Journal*, *8*(3), 868-878.

Tanaka, M., & Tatebe, O. (2012). Workflow scheduling to minimize data movement using multi-constraint graph partitioning [Conference Proceedings]. In *Proceedings of the 2012 12th ieee/acm international symposium on cluster, cloud and grid computing (ccgrid 2012)* (p. 65-72). IEEE Computer Society.

Tao, Q., Chang, H., Yi, Y., Gu, C., & Yu, Y. (2009). Qos constrained grid workflow scheduling optimization based on a novel pso algorithm [Conference Proceedings]. In *Gcc'09. eighth international conference on grid and cooperative computing, 2009.* (p. 153-159). IEEE.

Tao, Y., Jin, H., & Shi, X. (2007). Grid workflow scheduling based on reliability cost [Conference Proceedings]. In *Proceedings of the 2nd international conference on scalable information systems* (p. 12). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Tilak, S., & Patil, D. (2012). A survey of various scheduling algorithms in cloud environment [Journal Article]. *International Journal of Engineering Inventions*, *1*(2), 36-39.

Tolosana-Calasanz, R., BañAres, J. , Pham, C., & Rana, O. F. (2012). Enforcing qos in scientific workflow systems enacted over cloud infrastructures [Journal Article]. *Journal of Computer and System Sciences*, *78*(5), 1300-1315.

Topcuoglu, H., Hariri, S., & Wu, M.-y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing [Journal Article]. *IEEE Transactions on Parallel and Distributed Systems*, *13*(3), 260-274.

Tsai, C.-W., Huang, W.-C., Chiang, M.-H., Chiang, M.-C., & Yang, C.-S. (2014). A

hyper-heuristic scheduling algorithm for cloud [Journal Article]. *IEEE Transactions on Cloud Computing*, *2*(2), 236-250.

Tsai, C.-W., & Rodrigues, J. J. (2013). Metaheuristic scheduling for cloud: A survey [Journal Article]. *IEEE Systems Journal*, *8*(1), 279-291.

Tsai, C.-W., Song, H.-J., & Chiang, M.-C. (n.d.). A hyper-heuristic clustering algorithm [Conference Proceedings]. In *2012 ieee international conference on systems, man, and cybernetics (smc)* (p. 2839-2844). IEEE.

Van Hee, K. M. (2004). *Workflow management: models, methods, and systems* [Book]. The MIT press.

Varalakshmi, P., Ramaswamy, A., Balasubramanian, A., & Vijaykumar, P. (2011). An optimal workflow based scheduling and resource allocation in cloud [Book Section]. In *Advances in computing and communications* (Vol. 190, p. 411-420). Springer.

Verma, A., & Kaushal, S. (2017). A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. *Parallel Computing*, *62*, 1–19.

Viana, V., de Oliveira, D., & Mattoso, M. (2011). Towards a cost model for scheduling scientific workflows activities in cloud environments [Conference Proceedings]. In *2011 ieee world congress on services (services)* (p. 216-219). IEEE.

Wang, W.-J., Chang, Y.-S., Lo, W.-T., & Lee, Y.-K. (2013). Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments [Journal Article]. *The Journal of Supercomputing*, *66*(2), 1-29.

Wang, Y., & Lu, P. (2013). Dds: A deadlock detection-based scheduling algorithm for workflow computations in hpc systems with storage constraints [Journal Article]. *Parallel Computing*, *39*(8), 291-305. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0167819113000501`

Wickremasinghe, B., Calheiros, R. N., & Buyya, R. (n.d.). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications [Conference Proceedings]. In *2010 24th ieee international conference on advanced information networking and applications* (p. 446-452). IEEE.

Wieczorek, M., Hoheisel, A., & Prodan, R. (2008). Taxonomies of the multi-criteria grid

workflow scheduling problem [Book Section]. In (p. 237-264). Springer.

Wieczorek, M., Hoheisel, A., & Prodan, R. (2009). Towards a general model of the multi-criteria workflow scheduling on the grid [Journal Article]. *Future Generation Computer Systems*, *25*(3), 237-256.

Wieczorek, M., Prodan, R., & Fahringer, T. (2005). Scheduling of scientific workflows in the askalon grid environment [Journal Article]. *ACM SIGMOD Record*, *34*(3), 56-62.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*. Springer Science & Business Media.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., & Fisher, P. (2013). The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud [Journal Article]. *Nucleic acids research*, gkt328.

Wu, Q., Yun, D., Lin, X., Gu, Y., Lin, W., & Liu, Y. (2013). On workflow scheduling for end-to-end performance optimization in distributed network environments [Conference Proceedings]. In *Job scheduling strategies for parallel processing* (p. 76-95). Springer.

Wu, Z., Liu, X., Ni, Z., Yuan, D., & Yang, Y. (2013). A market-oriented hierarchical scheduling strategy in cloud workflow systems [Journal Article]. *The Journal of Supercomputing*, *63*(1), 256-293.

Wu, Z., Ni, Z., Gu, L., & Liu, X. (2010). A revised discrete particle swarm optimization for cloud workflow scheduling [Conference Proceedings]. In *2010 international conference on computational intelligence and security (cis)* (p. 184-188). IEEE.

Xing, B., & Gao, W.-J. (2014). Invasive weed optimization algorithm [Book Section]. In *Innovative computational intelligence: A rough guide to 134 clever algorithms* (p. 177-181). Springer.

Xu, M., Cui, L., Wang, H., & Bi, Y. (2009). A multiple qos constrained scheduling strategy of multiple workflows for cloud computing [Conference Proceedings]. In *2009 ieee international symposium on parallel and distributed processing with applications* (p. 629-634). IEEE.

Xue, S.-J., & Wu, W. (2012). Scheduling workflow in cloud computing based on hybrid particle swarm algorithm [Journal Article]. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *10*(7), 1560-1566.

Yan, C., Luo, H., Hu, Z., Li, X., & Zhang, Y. (2013). Deadline guarantee enhanced scheduling of scientific workflow applications in grid [Journal Article]. *Journal of Computers*, *8*(4), 842-850.

Yang, X., Wallom, D., Waddington, S., Wang, L., & Blower, J. D. (2014). Cloud computing in e-science: research challenges and opportunities [Journal Article]. *The Journal of Supercomputing*, *70*(1), 408-464.

Yang, Y., Liu, K., Chen, J., Liu, X., Yuan, D., & Jin, H. (2008). An algorithm in swindew-c for scheduling transaction-intensive cost-constrained cloud workflows [Conference Proceedings]. In *escience'08. ieee fourth international conference on escience, 2008.* (p. 374-375). IEEE.

Yassa, S., Chelouah, R., Kadima, H., & Granado, B. (2013). Multi-objective approach for energy-aware workflow scheduling in cloud computing environments [Journal Article]. *The Scientific World Journal*, *2013*, 1-13.

Yu, H., Bai, X., & Marinescu, D. C. (2005). Workflow management and resource discovery for an intelligent grid [Journal Article]. *Parallel Computing*, *31*(7), 797-811. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0167819105000670`

Yu, J., & Buyya, R. (2005). A taxonomy of scientific workflow systems for grid computing [Journal Article]. *Sigmod Record*, *34*(3), 44-49.

Yu, J., & Buyya, R. (2006a). A budget constrained scheduling of workflow applications on utility grids using genetic algorithms [Conference Proceedings]. In *Works'06. workshop on workflows in support of large-scale science, 2006.* (p. 1-10). IEEE.

Yu, J., & Buyya, R. (2006b). Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms [Journal Article]. *Scientific Programming*, *14*(3), 217-230.

Yu, J., Buyya, R., & Tham, C. K. (2005). Cost-based scheduling of scientific workflow applications on utility grids [Conference Proceedings]. In *First international conference on e-science and grid computing, 2005* (p. 139-147). IEEE.

Yu, J., Kirley, M., & Buyya, R. (2007). Multi-objective planning for workflow execution on grids [Conference Proceedings]. In *Proceedings of the 8th ieee/acm international conference on grid computing* (p. 10-17). IEEE Computer Society.

Yuan, D., Yang, Y., Liu, X., & Chen, J. (2010). A cost-effective strategy for intermediate data storage in scientific cloud workflow systems [Conference Proceedings]. In *2010 ieee international symposium on parallel and distributed processing (ipdps)* (p. 1-12). IEEE.

Yuan, D., Yang, Y., Liu, X., & Chen, J. (2011). On-demand minimum cost benchmarking for intermediate dataset storage in scientific cloud workflow systems [Journal Article]. *Journal of Parallel and Distributed Computing*, *71*(2), 316-332.

Yuan, D., Yang, Y., Liu, X., Zhang, G., & Chen, J. (2012). A data dependency based strategy for intermediate data storage in scientific cloud workflow systems [Journal Article]. *Concurrency and Computation: Practice and Experience*, *24*(9), 956-976.

Yuan, Y., Li, X., & Wang, Q. (2006). Time-cost tradeoff dynamic scheduling algorithm for workflows in grids [Conference Proceedings]. In *Cscwd'06. 10th international conference on computer supported cooperative work in design, 2006.* (p. 1-6). IEEE.

Yuan, Y., Li, X., Wang, Q., & Zhu, X. (2007). Cost optimization method for workflows with deadline constraints in grids [Conference Proceedings]. In *Cscwd 2007. 11th international conference on computer supported cooperative work in design, 2007.* (p. 783-788). IEEE.

Yuan, Y., Li, X., Wang, Q., & Zhu, X. (2009). Deadline division-based heuristic for cost optimization in workflow scheduling [Journal Article]. *Information Sciences*, *179*(15), 2562-2575.

Zeng, L., & Wang, Y. (2013). Optimization on content service with local search in cloud of clouds [Journal Article]. *Journal of Network and Computer Applications*, *40*, 1-10.

Zhang, S., Liu, Y., Wang, B., & Zhang, R. (2013). A novel resource allocation algorithm for a heterogeneous data center [Conference Proceedings]. In *2013 international conference on information science and applications (icisa)* (p. 1-4). IEEE.

Zhang, X., Niu, Y., Cui, G., & Wang, Y. (n.d.). A modified invasive weed optimization

with crossover operation [Conference Proceedings]. In *Intelligent control and automation (wcica), 2010 8th world congress on* (p. 11-14). IEEE.

Zhao, H., & Sakellariou, R. (2007). Advance reservation policies for workflows [Conference Proceedings]. In *Job scheduling strategies for parallel processing* (p. 47-67). Springer.

Zhao, L., Ren, Y., Li, M., & Sakurai, K. (2012). Flexible service selection with user-specific qos support in service-oriented architecture [Journal Article]. *Journal of Network and Computer Applications*, *35*(3), 962-973.

Zheng, W. (2010). *Explorations in grid workflow scheduling* (Thesis). University of Manchester.

Zheng, W., & Sakellariou, R. (2013). Budget-deadline constrained workflow planning for admission control [Journal Article]. *Journal of Grid Computing*, *11*(4), 1-19.

Zhu, M., Wu, Q., & Zhao, Y. (2012). A cost-effective scheduling algorithm for scientific workflows in clouds [Conference Proceedings]. In *2012 ieee 31st international performance computing and communications conference (ipccc)* (p. 256-265). IEEE.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Some of the research leading to this thesis has appeared previously in the following publications.

Journal Papers:

Ehab Nabiel Alkhanak, Sai Peck Lee, Saif Ur Rehman Khan, Cost-aware challenges for workflow scheduling approaches in cloud computing environments: Taxonomy and opportunities, Future Generation Computer Systems (2015), Volume 50, September 2015, Pages 3-21, ISSN 0167-739X, (One of the top 20 most downloaded papers for the last one year).

Ehab Nabiel Alkhanak, Sai Peck Lee, Reza Rezaei, Reza Meimandi Parizi. "Cost optimisation approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues", Journal of Systems and Software (2016), Volume 113, Pages 1-26, ISSN 0164-1212.

Conference Paper:

Ehab Nabiel Alkhanak, Sai Peck Lee, and Teck Chaw Ling. "A Hyper-Heuristic Approach using a Prioritized Selection Strategy for Workflow Scheduling in Cloud Computing.", The 4th International Conference on Computer Science and Computational Mathematics (ICCSCM 2015), SandKRS (Science and Knowledge Research Society), Pages 601-608, 2015.

Papers Presented:

Presenter in Postgraduate Research Excellence Symposium (PGReS) 2012, 26th September 2012, Organised by Faculty of Computer Science and Information Technology, University of Malaya.

Presenter in Postgraduate Research Excellence Symposium (PGReS) 2014, 15th May 2014, Organised by Faculty of Computer Science and Information Technology, University of Malaya.