Universidade Estadual de Campinas
Instituto de Computação

Thiago Augusto Lopes Genez

Providing Robustness to Non-robust Workflow Schedulers to the Uncertainties of Available Bandwidth

Provendo Robustez a Escalonadores de *Workflows* Sensíveis às Incertezas da Largura de banda Disponível

CAMPINAS
2017

Thiago Augusto Lopes Genez

Providing Robustness to Non-robust Workflow Schedulers to the Uncertainties of Available Bandwidth

Provendo Robustez a Escalonadores de *Workflows* Sensíveis às Incertezas da Largura de banda Disponível

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Dissertation presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Supervisor/Orientador: Prof. Dr. Edmundo Roberto Mauro Madeira
Co-supervisor/Coorientador: Prof. Dr. Luiz Fernando Bittencourt

Este exemplar corresponde à versão final da Tese defendida por Thiago Augusto Lopes Genez e orientada pelo Prof. Dr. Edmundo Roberto Mauro Madeira.

CAMPINAS
2017

Agência(s) de fomento e nº(s) de processo(s): FAPESP, 2012/02778-6

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

G287p

Genez, Thiago Augusto Lopes, 1987-
Provendo robustez a escalonadores de workflows sensíveis às incertezas da largura de banda disponível / Thiago Augusto Lopes Genez. – Campinas, SP : [s.n.], 2017.

Orientador: Edmundo Roberto Mauro Madeira.
Coorientador: Luiz Fernando Bittencourt.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Escalonamento de processos. 2. Computação em nuvem. 3. Fluxo de trabalho. 4. Sistemas distribuídos. I. Madeira, Edmundo Roberto Mauro, 1958-. II. Bittencourt, Luiz Fernando, 1981-. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Providing robustness to non-robust workflow schedulers to the uncertainties of available bandwidth
Palavras-chave em inglês:
Process scheduling
Cloud computing
Workflow
Distributed systems
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Edmundo Roberto Mauro Madeira [Orientador]
Carlos Alberto Kamienski
Paulo Romero Martins Maciel
Islene Calciolari Garcia
Leandro Aparecido Villas
Data de defesa: 13-12-2017
Programa de Pós-Graduação: Ciência da Computação

Universidade Estadual de Campinas
Instituto de Computação

Thiago Augusto Lopes Genez

Providing Robustness to Non-robust Workflow Schedulers to the Uncertainties of Available Bandwidth

Provendo Robustez a Escalonadores de *Workflows* Sensíveis às Incertezas da Largura de banda Disponível

Banca Examinadora:

- Prof. Dr. Edmundo Roberto Mauro Madeira (Presidente)
  Instituto de Computação (IC)/Universidade Estadual de Campinas (Unicamp)
- Prof. Dr. Carlos Alberto Kamienski
  Universidade Federal do ABC (UFABC)
- Prof. Dr. Paulo Romero Martins Maciel
  Universidade Federal de Pernambuco (UFPE)
- Profa. Dra Islene Calciolari Garcia
  Instituto de Computação (IC)/Universidade Estadual de Campinas (Unicamp)
- Prof. Dr. Leandro Aparecido Villas
  Instituto de Computação (IC)/Universidade Estadual de Campinas (Unicamp)

A ata da defesa com as respectivas assinaturas dos membros da banca encontra-se no processo de vida acadêmica do aluno.

Campinas, 13 de dezembro de 2017

To my father Carlos, to my mother Jandira and to my brother Lucas.

Success consists of going from failure to failure without loss of enthusiasm.

(Broadly attributed to Winston Churchill and Abraham Lincoln)

Agradecimentos – *Acknowledgement*

Seria impossível ter escrito essa Tese sem o suporte da minha família e dos meus amigos. Obrigado pelo apoio nas minha decisões para que eu pudesse completar mais uma importante etapa da minha vida. Vocês são tudo para mim. Obrigado por tudo!

Edmundo, muito obrigado por ter sido meu orientador. Sua orientação desde o mestrado foi essencial para que chegasse onde eu cheguei. Eu não tenho dúvida de que a experiência que eu obtive trabalhando junto com você nesta Tese, na dissertação de mestrado, nos artigos e nas disciplinas fará muita diferença na minha carreira. A você devo a confiança em minha capacidade como pesquisador. Não esquecerei dos seus ensinamentos e preciosos conselhos da sua inestimável confiança. Muito obrigado novamente! Espero que possamos continuar trabalhando juntos nos próximos anos.

Luiz, muito obrigado pela co-orientação e principalmente pela amizade e companheirismo. Desde o mestrado, as suas sugestões, a sua disponibilidade e o seu apoio foram essenciais para o desenvolvimento desta Tese. Eu também não tenho dúvida que a experiência que eu obtive trabalhando junto com você fará muita diferença da minha vida acadêmica. Aliás, de fato, já está fazendo. Muito obrigado! Agradeço também pelos papos nerds (e não nerds) de computeiro. Também espero que, junto com o Edmundo, possamos continuar trabalhando juntos nos próximos projetos de pesquisa.

Agradeço também ao Prof. Dr. Nelson L. S. da Fonseca pela colaboração no desenvolvimento desta pesquisa de doutorado. As suas sugestões, o seu apoio e a sua disponibilidade nos artigos foram extremamente essenciais para a construção tese. A experiência que eu obtive pela sua orientação nos artigos está fazendo diferença na formação da minha carreira acadêmica. Muito obrigado!

For those who contributed to this Thesis but did not speak Portuguese, I haven't forgotten you, guys! This acknowledgement also extends to Prof. Dr Rizos Sakellarious, for the opportunity to host and supervise me during my PhD sandwich at the University of Manchester, UK. Your supervision and collaboration were intrinsic to this doctoral research. Also, I would like to thank Prof. Dr. Posco Tso for the confidence to offer me the job position of Assistant Researcher at Loughborough University, UK. My eternal thanks for this opportunity that I can say that is kicking off my academic career. Thank you!

Muitíssimo obrigado a todos que passaram pelo Laboratório de Redes de Computadores (LRC) nesses últimos 5 anos! Aprendi muito com vocês. Também, a todos os amigos e amigas de Campinas, Londrina, Manchester e Loughborough que, de uma forma ou de outra, fizeram parte da minha vida nesses últimos 5 anos. Não irei citar nomes para não deixar ninguém de fora, porém todos estão implicitamente incluídos aqui e expresso a minha profunda gratidão por fazer parte da minha felicidade.

Aos demais mestrados, doutorandos, funcionários e professores do Instituto de Computação (IC) que colaboraram diretamente e indiretamente com a minha formação, e se esforçaram para que eu pudesse desenvolver meu trabalho com o melhor apoio possível. Também não irei citar nomes para não ser injusto com ninguém, porém já deixo registrado

a minha profunda gratidão por vocês.

Muito obrigado aos melhores pais e ao melhor irmão do mundo por existirem e por serem a minha família! Muitíssimo obrigado pela compreensão durante minha ausência nestes últimos anos.

Thank you, Jane, for your support, for the difficult times, for the pleasant moments, for the cares, for the trips, for the laughter, for the companionship, for the friendship, for the encouragement and for all the other indescribable things we enjoyed during our time together. Thank you! Lo siento mucho por Cuba. La Habana, aquí vamos en 2018!

Por último mas não menos importante, eu não poderia deixar de agradecer ao povo brasileiro, principalmente ao povo do Estado de São Paulo. A bolsa de pesquisa da FAPESP (2012/02778-6) a sua taxa de bancada ajudaram para que eu permanecesse com o foco em minha pesquisa durante esses anos de doutorado. Também agradeço à FAPESP (2014/08607-4) pelo suporte financeiro por viabilizar o doutorado sanduíche realizado na Universidade de Manchester, uma etapa fundamental para o desenvolvimento desta pesquisa de doutorado. Muito obrigado!

Agradecer a todos que me ajudaram a desenvolver esta Tese é uma tarefa árdua. O principal problema de um agradecimento seletivo não é decidir quem incluir, mas decidir quem não mencionar. Se nestes agradecimentos não há nenhuma referência direta à você que está lendo-os, e você tem motivos claros que contribuíram com este trabalho, então, primeiramente, peço-lhe desculpas e, em seguida, permito-lhe escrever de próprio punho seu nome nas linhas em branco seguintes.

Resumo

Para que escalonadores de aplicações científicas modeladas como *workflows* derivem escalonamentos eficientes em nuvens híbridas, é necessário que se forneçam, além da descrição da demanda computacional desses aplicativos, as informações sobre o poder de computação dos recursos disponíveis, especialmente aqueles dados relacionados com a largura de banda disponível. Entretanto, a imprecisão das ferramentas de medição fazem com que as informações da largura de banda disponível fornecida aos escalonadores difiram dos valores reais que deveriam ser considerados para se obter escalonamentos quase ótimos. Escalonadores especialmente projetados para nuvens híbridas simplesmente ignoram a existência de tais imprecisões e terminam produzindo escalonamentos enganosos e de baixo desempenho, o que os tornam sensíveis às informações incertas.

A presente Tese introduz um procedimento pró-ativo para fornecer um certo nível de robustez a escalonamentos derivados de escalonadores não projetados para serem robustos frente às incertezas decorrentes do uso de informações imprecisas dadas por ferramentas de medições de rede. Para tornar os escalonamentos sensíveis às incertezas em escalonamentos robustos às essas imprecisões, o procedimento propõe um refinamento (uma deflação) das estimativas da largura de banda antes de serem utilizadas pelo escalonador não robusto. Ao propor o uso de estimativas refinadas da largura de banda disponível, escalonadores inicialmente sensíveis às incertezas passaram a produzir escalonamentos com um certo nível de robustez a essas imprecisões. A eficácia e a eficiência do procedimento proposto são avaliadas através de simulação. Comparam-se, portanto, os escalonamentos gerados por escalonadores que passaram a usar o procedimento proposto com aqueles produzidos pelos mesmos escalonadores mas sem aplicar esse procedimento. Os resultados das simulações mostram que o procedimento proposto é capaz de prover robustez às incertezas da informação da largura de banda a escalonamentos derivados de escalonardes não robustos às tais incertezas. Adicionalmente, esta Tese também propõe um escalonador de aplicações científicas especialmente compostas por um conjunto de *workflows*. A novidade desse escalonador é que ele é flexível, ou seja, permite o uso de diferentes categorias de funções objetivos. Embora a flexibilidade proposta seja uma novidade no estado da arte, esse escalonador também é sensível às imprecisões da largura de banda. Entretanto, o procedimento mostrou-se capaz de provê-lo de robustez frente às tais incertezas.

É mostrado nesta Tese que o procedimento proposto aumentou a eficácia e a eficiência de escalonadores de *workflows* não robustos projetados para nuvens híbridas, já que eles passaram a produzir escalonamentos com um certo nível de robustez na presença de estimativas incertas da largura de banda disponível. Dessa forma, o procedimento proposto nesta Tese é uma importante ferramenta para aprimorar os escalonadores sensíveis às estimativas incertas da banda disponível especialmente projetados para um ambiente computacional onde esses valores são imprecisos por natureza. Portanto, esta Tese propõe um procedimento que promove melhorias nas execuções de aplicações científicas em nuvens híbridas.

Abstract

To derive efficient schedules for the tasks of scientific applications modelled as workflows, schedulers need information on the application demands as well as on the resource availability, especially those regarding the available bandwidth. However, the lack of precision of bandwidth estimates provided by monitoring/measurement tools should be considered by the scheduler to achieve near-optimal schedules. Uncertainties of available bandwidth can be a result of imprecise measurement and monitoring network tools and/or their incapacity of estimating in advance the real value of the available bandwidth expected for the application during the scheduling step of the application. Schedulers specially designed for hybrid clouds simply ignore the inaccuracies of the given estimates and end up producing non-robust, low-performance schedules, which makes them sensitive to the uncertainties stemming from using these networking tools.

This thesis introduces a proactive procedure to provide a certain level of robustness for schedules derived from schedulers that were not designed to be robust in the face of uncertainties of bandwidth estimates stemming from using unreliable networking tools. To make non-robust schedulers into robust schedulers, the procedure applies a deflation on imprecise bandwidth estimates before being used as input to non-robust schedulers. By proposing the use of refined (deflated) estimates of the available bandwidth, non-robust schedulers initially sensitive to these uncertainties started to produce robust schedules that are insensitive to these inaccuracies. The effectiveness and efficiency of the procedure in providing robustness to non-robust schedulers are evaluated through simulation. Schedules generated by induced-robustness schedulers through the use of the procedure is compared to that produced by sensitive schedulers. In addition, this thesis also introduces a flexible scheduler for a special case of scientific applications modelled as a set of workflows grouped into ensembles. Although the novelty of this scheduler is the replacement of objective functions according to the user's needs, it is still a non-robust scheduler. However, the procedure was able to provide the necessary robustness for this flexible scheduler be able to produce robust schedules under uncertain bandwidth estimates.

It is shown in this thesis that the proposed procedure enhanced the robustness of workflow schedulers designed especially for hybrid clouds as they started to produce robust schedules in the presence of uncertainties stemming from using networking tools. The proposed procedure is an important tool to furnish robustness to non-robust schedulers that are originally designed to work in a computational environment where bandwidth estimates are very likely to vary and cannot be estimated precisely in advance, bringing, therefore, improvements to the executions of scientific applications in hybrid clouds.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

Today, workflows are frequently used to model large-scale distributed scientific applications, composed of numerous interconnected computational tasks [129]. A workflow facilitates the expression of multi-step task workloads in a manner which is easy to understand, debug, and maintain [41]. It is used to analyse large amounts of data and to run complex scientific simulations and experiments efficiently [118]. By using workflows as a tool to model scientific applications, researchers and scientists can collaborate on designing a single large-scale distributed application. A workflow is considered distributed if its tasks are more loosely coupled, for example, through files, and therefore can be executed on geographically distributed systems, such as clusters, clouds, and grids, or multiple computational facilities and/or scientific instruments at user facilities [56]. This thesis focuses on studies of algorithms to orchestrate the execution of scientific workflow in cloud computing environments.

Cloud computing is a distributed computing paradigm that has now achieved worldwide scale adoption. It provides storage and processing capabilities on-demand by relying on high-capacity data centres accessible through the Internet, allowing users to access their data and/or applications anywhere an Internet connection is available [30, 133]. Among the main advantages that has made cloud computing popular are the upfront investment avoidance and abstraction of technical details for the user. These are results of a virtualised environment offered though interfaces at different abstraction levels, commonly named *Something as a Service* [7, 80, 147], which are canonically summarised into three service models: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). For instance, cloud customers can use/run applications from an SaaS cloud provider; or develop and deploy their own applications on a platform provided by a PaaS cloud provider; or extend their available in-house computing power by leasing resources in terms of virtual machines (VMs) from an IaaS cloud provider. Particularly, this last example represents the so-called hybrid cloud [34, 67–70], a composition of an infrastructure that jointly comprises resources available at user facilities (a private cloud) with VMs leased from IaaS cloud providers (a public cloud).

Figure 1.1 illustrates the elasticity provided by cloud computing in a private cloud. The red line represents the computational demand of a private cloud, while the blue line shows the traditional way of provisioning internal computing resources, which basically consists of buying new physical machines. As the classical way does not provide elasticity,

overcapacity may occur during non-peak times (shaded area in blue), which implies in waste of resources (and money). On the other hand, during peak times, some users may not be served because the available private resources are fully busy or may not be sufficient to meet the required computing demand, resulting in sub-capacity issues (shaded area in red). Finally, the green line shows the on-demand resource provisioning offered by cloud computing. As cloud computing provides elasticity, the computing capacity of a private cloud can easily follow the computational demand, leasing resources in peak times and releasing them in non-peak times, for instance.

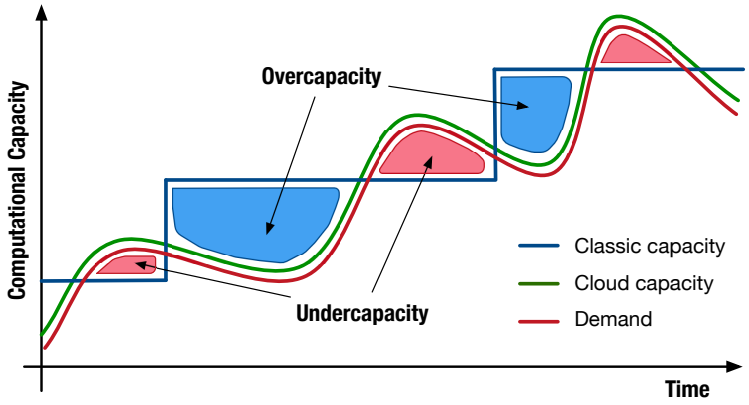


Figure 1.1: Illustration of elasticity offered by cloud computing to private cloud owners.

Traditionally, large-scale scientific workflows were widely executed on dedicated high performance supercomputers such as private campus clusters or cyberinfrastructure systems [33, 84]. These systems were used to be statically partitioned, expensive, as well as inefficient in adapting to the surge of demand. However, with the emergence of cloud computing, the scientific community regards it as a potentially attractive source of low-cost computing resources [102]. Instead of using a dedicated supercomputer, which typically offers best-effort quality of service, clouds give more flexibility in creating a controlled and managed computing environment to offload workflow executions [100]. Clouds offer an easily accessible, flexible, and scalable infrastructure for the deployment of large-scale scientific workflows. They enable Workflow Management Systems (e.g., Pegasus [1] [58]) to access a compute infrastructure on-demand and on a pay-per-use basis, facilitating the life of scientists to execute their scientific workflows [107, 147]. This is done by leasing virtualised compute resources (VMs) with a predefined CPU, memory, storage, and bandwidth capacity to join the available resources at users premises [61, 102, 137]. Actually, different VM flavours are available at varying prices to suit a wide range of application needs. Using a cloud-based infrastructure on-demand and follow the computing demand, VMs can be elastically leased and released and they are charged per time frame, or billing period [107, 118].

Indeed, scientists can execute most of their workflows (on a hybrid cloud) using their own computing infrastructure (private cloud), and yet lease resources from a cloud provider (public cloud) on demand to offload some extra workloads [26, 34]. It was reported that 48 percent of U.S. government agencies have moved at least one workflow

[1] `https://pegasus.isi.edu/`

to a cloud provider following the federal cloud computing strategy published in February 2011 [34]. It is also reported that the US Department of Energy (DOE) has been offloading scientific workflows to be executed on a cloud-based distributed-area system because it gives more flexibility in creating a controlled and managed computing environment to execute these large-scale applications [56]. Despite these advantages that the cloud brings, the process of offloading workflow's workloads to hybrid cloud poses some fundamental questions that should be answered before starting renting resources [34, 67, 68], such as:

- Which type of VM flavour should be leased to speedup the execution but with a budget constraint?
- Which tasks of the workflow should be offloaded to be executed on the leased VM in order to satisfy deadline constraints?

These questions can be answered by a workflow scheduler, a fundamental component of a decision-making process in distributed computing systems [112]. The scheduler is the element responsible for determining which task of the workflow will run on what virtual machines, distributing the workflow's tasks to different processing elements so that they can run in parallel, speeding up the execution of the application. These questions are answered considering the information about the workflow as well as information about the target system, including the processing capacity of each resource and the available capacities of communication links connecting the resources. By using this information as input data, the scheduler performs the decision-making process according to an objective function, which can have one or more criteria to be optimised, such as the schedule length (makespan), communication and computing costs, resource utilisation, fairness, and so forth. Due to this, the execution time and monetary cost of running a set of computational tasks on a pay-per-use basis distributed system are strongly dependent on the scheduler's decisions.

Most workflow schedulers generally assume that all input data is known in advance. They are called deterministic because they make this assumption. However, such assumption is not always true, and the scheduler ends up producing poor schedules that can degrade the workflow execution. As they ignore the (natural) inaccuracy of the input data, the produced schedules are considered poor because they are non-robust under these inaccuracies. A schedule is said to be *robust* if it is able to absorb (or be *insensitive* to) the uncertainty in the input data to guarantee that the performance of the workflow execution will experience a low or no degradation. Examples of workflow's performance degradation caused by the use of poor schedules include an unexpected increase in cost as well as in workflow execution time. Specially, this thesis focuses on the use of uncertain information about the available bandwidth by deterministic workflow schedulers that were not designed to cope with inaccurate bandwidth information. Therefore, this thesis attempts to mitigate the production of non-robust (poor) schedules by deterministic scheduler even in the presence of a high bandwidth variability – which is very likely to occur during the execution of the workflow [121, 122] because bandwidth is a shared resource. The question of how to make deterministic schedulers in yielding robust schedules under imprecise bandwidth estimates is addressed in this thesis. The terms *robust* and *insensitive* will be used synonymously hereinafter.

Uncertainties on the bandwidth availability can impact delays since Internet links, whose available bandwidth fluctuates considerably because of the shared communication infrastructure, are used to transmit data between private and public clouds. Although virtualisation isolates the machine in clouds, ensuring low variabilities in the description of the machine CPU-processing power [147], the bandwidth of inter-cloud and intra-cloud links are shared without isolation, which can experience high variability and potentially create bottlenecks on the executions of workflows. These network bottlenecks are mainly caused by the use of poor schedules generated using inaccurate bandwidth information. Inaccurate bandwidth information needs to be addressed in advance so that deterministic schedulers can be able to mitigate the production of poor (non-robust) schedules. Understanding the impact of network delays on scheduling decisions caused by the use of imprecise bandwidth information is thus fundamental for cloud resource provisioning, and we address this topic in this thesis.

In order not to get rid of existing non-robust deterministic workflow schedulers for hybrid clouds and enhancing them with a certain level of robustness, this thesis proposes a proactive procedure (mechanism) for this purpose. The procedure does not change the scheduling algorithm or the scheduler itself, it is a procedure that assists the scheduler in producing robust schedules under imprecise bandwidth estimations. The proposed procedure deflates the estimated available bandwidth before furnishing it to the scheduler. Instead of using imprecise bandwidth measurements directly provided by monitoring networking tools [14, 126], deterministic schedulers will use as input data a deflated value of the original measurements in an attempt to *proactively* mitigate the production of poor schedules. The deflating factor is computed by the proposed procedure through a multiple linear regression using historical information of previous workflow executions. Simulations are used to show the advantage of the proposed procedure in furnishing robustness for existing non-robust schedulers in the literature under uncertainties of available bandwidth. Results obtained from simulations show that, when non-robust deterministic schedulers employed the proposed procedure, they produced better (robust) schedules than those produced ignoring the uncertainties considered. The terms *procedure* and *mechanism* will be synonymously used hereinafter.

This thesis also proposes a novel flexible scheduler for workflow ensembles, a type of large-scale scientific applications that comprises a set of inter-related workflows instead of being composed using one workflow only. Workflows in an ensemble typically have a similar structure, but they differ in their input data, number of tasks, and individual task sizes. That is the reason that they are called "inter-related" workflows. This scheduler is considered *flexible* because it allows the replacement of the objective function, making the scheduler usable in various scenarios. In this sense, the user can customise a objective function according to his/her needs, such as to minimise the workflow execution cost, maximise resource utilisation, or minimise the workflow execution time. Simulation results show that the flexibility of the scheduler has been achieved since each function could produce schedules that satisfied its corresponding objective. Although the novelty of this scheduler is this flexibility to allow the substitution of the objective function, it is still a non-robust scheduler under uncertainties of available bandwidth once it provides schedules without addressing the imprecision of input data. However, according to results

obtained from simulations, the proposed procedure was also able to furnish robustness for our flexible scheduler under imprecise bandwidth estimates as well. By employing the proposed procedure, the flexible scheduler has produced robust schedules for workflow ensembles in the presence of uncertain bandwidth estimates since the available bandwidth in links may vary and cannot be estimated precisely in advance by monitoring networking tools.

The following sections detail this thesis and are organised as follows. Section 1.1 lists the contributions, while Section 1.2 summarises all publications resulting from this thesis. Section 1.3 finalises this introductory chapter detailing the content of the next chapters.

1.1 Contributions

The list below describes all the contributions of this thesis:

- A proposal of a procedure to reduce the impact of the uncertainties of available bandwidth on the schedule generated by deterministic scheduling algorithms. The proposed procedure aims to furnish a certain level of robustness to the uncertainties of available bandwidth for sensitive (non-robust) workflow schedulers that were not designed to cope with such uncertainties. Results obtained from simulations show that, when sensitive workflow schedulers have employed the proposed procedure, they have produced better (robust) schedules than those produced ignoring the uncertainties considered. These experiments demonstrate the advantage of the mechanism in "robustify"[2] these non-robust workflow schedulers under uncertainties of available bandwidth;

- A proposal of a flexible scheduler for workflow ensembles that allows the customisation of the scheduler's objective function according to the user's needs. Three types of objective function were evaluated: minimisation of the makespan, maximisation of fairness, and minimisation of costs. Results obtained from simulations show that the flexibility of the proposed scheduler has been achieved since each function could produce schedules that satisfied its corresponding objective. These experiments demonstrate that the feature of flexibility in replacing the scheduler's objective function extends the usefulness of the scheduler in various scenarios; and

- A proposal of a robust scheduler for workflow ensembles under uncertainties of available bandwidth. Results of simulations considering diverse scenarios, based on several degrees of uncertainty in available bandwidth estimates, characteristics of bandwidth estimations and ensemble applications, demonstrate the advantages of the robust flexible scheduler in a cloud-based scenario.

[2]The term *robustify* does not officially exist in dictionaries of the English language. It is a neological verb that this thesis employs to describe an action of making a non-robust scheduler into a robust one, which is more tolerant to unexpected events in the context of this thesis.

1.2 Publications

The results presented in this thesis generated 4 publications. Nevertheless, these publications were instigated thanks to 4 previous publications carried out in the early stages of this PhD. The mechanism to provide robustness for deterministic non-robust schedulers be able to produce robust schedules under the uncertainty of available bandwidth was published in [67] and extended in [68]. Both publications are meticulously discussed in Chapter 3, and they were outcome of a simple question raised from a previous publication [70]: *how the variability in available bandwidth in inter-cloud links may affect the execution cost of applications?* In the meantime, on the other side of this study, we were researching the scheduling of several inter-related workflows grouped into an ensemble, since all efforts so far were directed towards the scheduling problem of scientific applications represented by a single workflow only. Due to this, a flexible scheduler for workflow ensembles was published in [73], and a research joining that mechanism to robustify this flexible scheduler was published in [74]. Both publications are meticulously discussed in Chapter 4. Analogously, these publications were outcome of the previous publications [75], [72], [71], and [66]; in which all of them tackle the workflow scheduling problem using a single workflow. Therefore, the whole scope of the study conducted in this PhD was based on a total of 8 publications, summarising 6 international conference papers and 2 international journal articles.

1.3 Thesis Overview

The thesis is organised as follows:

Chapter 2 introduces the background and key concepts needed to understand the content of this thesis. The definition of the cloud computing was presented, focusing on hybrid clouds for the development of this thesis. Scientific applications modelled as workflows of tasks and represented by direct acyclic graphs (DAGs) were also described, including their characteristic regarding the loosely-coupled tasks stitched by data- and control-flow dependencies through files. This chapter also introduces the general definition of the workflow scheduling problem and the main characteristics of a deterministic workflow scheduler, including the assumption that all input data is known with certainty in advance. However, this assumption is not always true, and the schedules can be negatively affected during the execution of the workflow. The chapter finalises posing the importance of considering the accuracy of input data of a deterministic workflow scheduler is presented, especially that information related to the available bandwidth, serving as a motivation for the focus given in the experiments carried out in the following chapters.

Chapter 3 presents the proposed procedure to provide robustness for deterministic workflow schedulers under uncertainties of available bandwidth. This chapter highlights that uncertainties of available bandwidth can be a result of imprecise measurement and monitoring network tools and/or their incapacity of estimating the real value of the available bandwidth during the application scheduling time. As bandwidth estimations will be given in ranges rather than deterministic values, the schedules produced by deterministic schedulers may make decisions that lead to ineffective executions of applications. Due to

this, these schedules are called non-robust schedules under uncertainties of bandwidth estimates since they are insensitive to these inaccuracies. Therefore, robust schedulers that are able to identify these imprecisions can provide a robust schedule that is insensitive to these uncertainties, tolerating variations in bandwidth estimates up to a certain degree of uncertainty. Simulations are used to show the advantage of the proposed procedure in this chapter, which was able to "robustify" existing non-robust schedulers in the face of uncertainties naturally embedded in bandwidth estimates furnished by monitoring/measurement networking tools.

Chapter 4 presents the flexible scheduler for workflows in an ensemble, a type of scientific applications that comprises a set of inter-related workflows. Workflows in an ensemble typically have a similar structure, distinguished only by their sizes (number of tasks), input data, and individual computational task sizes. We consider that a workflow scheduler is *flexible* when it allows the replacement of the objective function according to the user's needs. Similar to a modular design, the objective function can be seen as a "module" that can be switched when necessary, making the scheduler usable in various scenarios. Simulation results show that the flexibility of the proposed scheduler has been achieved since each function could produce schedules that satisfied its corresponding objective. Although the novelty of this scheduler is this flexibility to allow the substitution of the objective function, it is still a non-robust scheduler under uncertainties of available bandwidth since it provides schedules without addressing the imprecision of input data. Therefore, this chapter also includes the applicability of the procedure, presented in Chapter 3, to make this flexible non-robust scheduler into a robust flexible scheduler under uncertainties of available bandwidth. Results of simulations considering diverse scenarios, based on several degrees of uncertainty in available bandwidth estimates, characteristics of bandwidth estimations and workflow applications, demonstrate the advantages of the robustified-flexible scheduler.

Chapter 5 finalises this thesis by presenting a discussion of the conclusion extracted from this research and proposes suggestions for future work.

Chapter 2

Background and Key Concepts

In this chapter we introduce the background and key concepts needed to understand the thesis, besides general definitions used in this study. Section 2.1 presents the definition of cloud computing, while Section 2.2 describes an overview of scientific applications that are modelled as non-cyclic workflows of tasks. Section 2.3 presents the general definition of the workflow scheduling problem, while Section 2.4 finalises this chapter by presenting a summary that relates the content presented in this chapter with the next chapters of this thesis. Definitions and notations specific to the proposed algorithms are introduced when they are presented.

2.1 An Overview of Cloud Computing

Cloud computing is a distributed computing paradigm that has now achieved wide scale adoption. It is one of the biggest buzzword that is attracting attention from both academy and IT industry across the world. It provides storage and processing capabilities on-demand by relying on high-capacity data centres accessible through the Internet. A cloud-based environment system allows, for instance, users to access their data and/or applications anywhere an Internet connection is available [8,80,133,134]. Among the main advantages that has made cloud computing so popular today are the upfront investment avoidance, abstraction of technical details for the user, the lowering of their operating and maintenance costs, and the scalability provided by on-demand model in a pay-per-use basis [147], usually charged for hours of use.

The National Institute of Standards and Technology (NIST) proposed the following definition of cloud computing [99]: *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"*. Although there are various definitions of cloud computing in the literature [134], this thesis adopts the NIST's definition in [99].

In cloud computing, resources are results of a virtualised environment offered though interfaces at different abstraction levels, commonly named as "*Something* as a Service" [23]. Thanks to the enhancement of virtualisation technology, cloud computing employs a

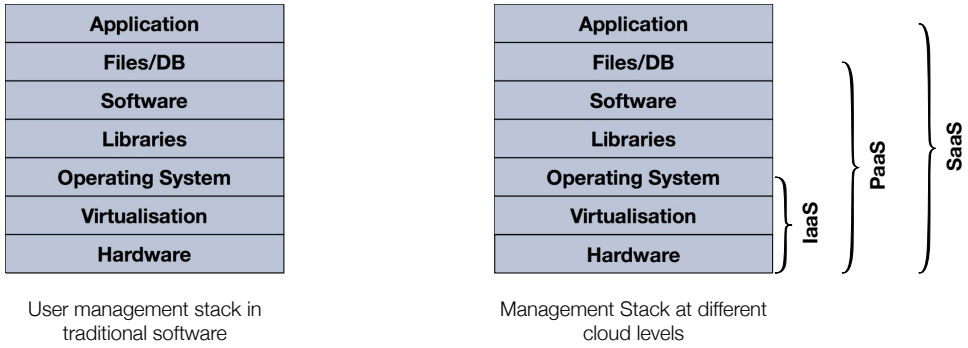


Figure 2.1: Service models in cloud computing and their level of management (from [23]).

service-driven business model. It is is canonically summarised into mainly three service-based models [99]: IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service).

The management efforts as compared to usual software deployment outside the cloud is shown in Figure 2.1 and discussed bellow:

- *Infrastructure as a Service*: At the bottommost level of abstraction, IaaS offers virtual machines as a service, which means the user can lease a "compute" that is remotely accessible with administrative privileges. In this model, VMs can be chosen from a set of configurations offered by the IaaS provider, specifying CPU speed (frequency), amount of RAM, amount of disk space, I/O speed, network connection speed, etc. Users, however, cannot access and manage physical machines (within provider's data centers) where these VMs run. Using this model, for instance, users can expand their computational power as necessary, since VMs can be leased and released on-demand and with the desired hardware configurations. In practice, this generally implies lower upfront capital investment because one does not need to acquire physical servers to fulfil peaks in demand (which, by definition, are likely to be rare events), which can be fulfilled by renting VMs as the demand grows. When utilising an IaaS provider, the user can be charged according to the VM configuration chosen and duration of the rental. Amazon Elastic Compute Cloud (EC2 [1]), ElasticHosts[2], and Google Compute Engine[3] are well-known examples of IaaS providers in the today's market;

- *Platform as a Service*: At the middle level, PaaS offers a platform-layer of development framework for the cloud user (software developer) to develop and deploy his/her own application. The user has control over the application features and development, but has no control over the physical infrastructure, and the development framework/tools/software is offered by the PaaS provider. Unlike the users of a IaaS provider, who get full control of the entire VM (from the operating system to all the applications running on top of it), users of a PaaS have there limited control this time, focused only in their applications. Using this model, the user has

[1] `https://aws.amazon.com/ec2/`
[2] `https://www.elastichosts.com`
[3] `https://cloud.google.com/products/compute-engine/`

to focus only in the application's development, while the underlying infrastructure is available transparently during the application's deployment and execution. A well-known example of a PaaS provider is the Google Apps Engine[4] and Docker[5]; and

- *Software as a Service*: At the topmost level, SaaS offers on-line software deployments developed by providers and accessible by users through the network, who can utilize software features made available by the developer and/or change pre-defined configurations established by the provider. Using this model, the user cannot manage or control the underlying platform where the application is deployed and executed. A SaaS provider typically provides therefore applications ready-to-use for end-users. Examples of SaaS providers include Google Apps for Business[6], Salesforce.com[7], Dropbox[8], ShareLaTeX editor[9], besides the recent services of audio and video streaming, such as Netflix[10]and Spotify[11].

The NIST also classifies a cloud provider according to the modelling of resource (service) deployment. All of them, however, can provision resources as IaaS, PaaS, or/and SaaS. The NIST's classification comprises in four main models [99]:

- *Private cloud* or *internal cloud*: The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units or campus of a university). It may be owned, managed, and operated by the organization, a third-party, or some combination of them, and it may exist on or off premises. To build a private cloud, however, the owner can not avoid up-front investment and maintenance cost related to the operational expenditure (OPEX) computational resources. The owner carries the same staffing, management, maintenance and capital expenses as traditional data center ownership. Additional private cloud expenses may include virtualisation, cloud software and cloud management tools. Nevertheless, implementing a private cloud model can improve the allocation of resources within an organisation by ensuring that the availability of resources to individual departments/business functions can directly and flexibly respond to their demand. Moreover, it makes use of the computing resource more efficient than traditional LANs and can also contribute to the redution of organisation's carbon footprint [147];

- *Community cloud*: Similar to the provide cloud, the cloud infrastructure in a community cloud is provisioned for exclusive use within a specific community of consumers from organizations that have shared concerns (e.g., security requirements, policies, jurisdiction, etc.). A community cloud is therefore a collaborative effort in

[4] `https://appengine.google.com`
[5] `https://www.docker.com/`
[6] `http://www.google.com/enterprise/apps/business/`
[7] `http://www.salesforce.com/`
[8] `https://www.dropbox.com/`
[9] `https://www.sharelatex.com/`
[10] `https://www.netflix.com/`
[11] `https://www.spotify.com`

which infrastructure is shared between several organizations from a specific community with common concerns. It may be owned, managed, and operated by one or more of the organizations in the community or by a third-party and hosted internally or externally. Community clouds are often designed for businesses and organizations working on joint projects, applications, or research, which requires a central cloud computing facility for building, managing and executing such projects, regardless of the solution rented [39];

- *Public (Commercial) cloud*: It is a cloud where the resources are provisioned as services to the general public. The services are often provided on a pay-per-use basis, usually charged for hours of use. However, a very-basic service can be provided for free as a complimentary trial service to the user test the provider's offered services. Google Cloud Platform[12], Google App Engine[13] and G-Suite[14] are examples of public cloud providers owned by Google that provide, respectively, IaaS, PaaS, and SaaS to the general public; and

- *Hybrid cloud*: It is a cloud that is composed by a combination of public and private (or community) clouds in order to address the limitations of each approach. In a hybrid cloud, part of the services may run in the private or community resources, while some parts may run in resources leased from public cloud providers. Whilst private clouds do offer a certain level of scalability (depending the size of the local data center), public cloud services will offer scalability with fewer boundaries because resource is pulled from the larger cloud infrastructure. By moving as many non-sensitive functions as possible to the public cloud, it allows an organisation to benefit from public cloud scalability whilst reducing the demands on a private cloud. Non-critical information, for example, can be processed in public cloud resources, while sensitive and critical are computed into private or community resources. The private cloud element of the hybrid cloud model not only provides the security where it is needed for sensitive operations, but can also satisfy regulatory requirements for data handling and storage where it is applicable. On the other hand, public clouds are likely to offer more significant economies of scale, and so greater cost efficiencies, than private clouds. Hybrid clouds, therefore, allow organisations to access these savings for as many non-sensitive operations as possible whilst still keeping sensitive computations secure[15].

In relation to the public IaaS cloud provider, the user must take into account the main types of rental models practised by the most IaaS providers in the today's market. In general, the charging model of VM renting are usually specified in a Service Level Agreement (SLA) that defines how VM rentals are charged and which conditions apply to the specified rental. Popular models include [23]:

[12] `https://cloud.google.com/appengine/`
[13] `https://cloud.google.com/appengine`
[14] `https://gsuite.google.com/`
[15] `http://www.interoute.com/what-hybrid-cloud`

- *On-demand*: The cloud user simply chooses the VM to be rented according to his/her needs and is charged as long as the VM is still running. Using this type of contract, the user can stop the VM rental at any time without paying any fee;

- *Reserved*: In this contract model, the user pays a fixed fee for the right to rend on-demand VM for a specific amount of time (days, weeks, or even months). The rental price using this model is usually lower than the prices through the on-demand rental agreement. Using this type of long-term contract, the VM will normally be available until the end of the agreement, even if the user is not using it;

- *Spot*: It is another type VM rental contracts where VM prices are based on the market offer/demand, and the user establishes a maximum price (bid) he/she is willing to pay for that type of VM. The user can have this VM as long as the price does not surpasses the established bid, and if it does, the VM can be interrupted by the provider without user acknowledgement. Spot VMs are often available at a discount compared to on-demand pricing, the user can significantly reduce the cost of running his/her applications, grow his/her application's compute capacity and throughput for the same budget; and

- *Auctions*: It is a not very common type of rental contracts where VMs are auctioned based on demand – with the price reflecting demand for particular types of VMs. This type of market model is particularly prevalent when specialist VMs (e.g. with GPUs or supporting specialist software libraries) that are made available to users.

Before renting VMs, the user must take into account which type of charging model will satisfy his/her application execution's needs to not compromise the available budget, which sometimes might be tight. In general, on-demand VMs are more expensive than reserved VMs, and spot VMs are cheaper due to the risk of interruption. Indeed, the charging model to be chosen is highly dependent on the application requirements and demand predictability. For example, if the user knows a VM will be needed for a period of one year and it cannot be interrupted, this VM could be leased through reserved contract model. On the other hand, if the VM is needed for a period of time (e.g. a few hours or days), on-demand VMs can result in lower bills due to reduced upfront costs. However, if an application (or VM unavailability) can be interrupted, a spot contract would be more interesting to be signed since it would suffice and result in lower bills.

The today's proliferation of commercial cloud computing providers has generated significant interest in the scientific community [54, 88]. The cloud computing paradigm suits well a variety of applications, specially scientific ones modelled as workflows that require ubiquitous data-processing availability [23, 87, 129]. It has gained popularity because it offers users several options and benefits compared to the traditional high-performance computing (HPC) system, especially when it comes to provisioning on-demand resources. Instead of using a dedicated supercomputer, which typically offers expensive, higher-quality, better-performing computing services, clouds offer more flexibility in creating a manageable, customisable, low-cost computing environment system [78, 85]. Cloud computing offers, therefore, new opportunities for scientists to conduct in a quick and cheap

way computing-intensive experiments through scientific applications [51, 86, 100]. New opportunities also brings new challenges, as this thesis focuses.

Indeed, as scientific data volumes grow, scientists will increasingly require data processing services that support easy access to distributed computing platforms such as clouds [56, 86]. Because scientific applications often require resources beyond those available on a scientist's desktop, the provisioning of computers (virtual machines) on-demand through the Internet has made a scientist's life easier in making computing-hungry scientific experiments, since they have the option of quickly renting entire virtual machines at any time from anywhere on a pay-per-use basis [20, 107, 147].

2.2 An Overview of Scientific Workflows

Scientific workflows are a useful tool for orchestrating data intensive applications in a broad range of domains, such as astronomy, bioinformatics, climate science, and others [20, 54, 85, 86, 88, 129]. They stitch together loosely-coupled computational tasks connected by data- and control-flow dependencies on behalf of the scientists who attempts to conduct complex large-scale scientific experiments using computing-intensive applications [85]. In fact, workflows enable scientists to easily define the composition of complex scientific applications as parallel computational and data processing pipelines. Management, analysis, simulation, and visualization of scientific data [87] are examples of computational tasks presented in these scientific (e-Science) applications [12, 53].

The concept of workflow has its roots in commercial enterprises as a business process modelling tool [12]. These business workflows aim automatise and optimise the processes of an organization, seen as a sequence of activities [118]. Due to this, the notion of business workflow has quickly introduced in science to support the modelling of large-scale scientific applications [129, 142]. Established as an important abstraction tool, scientific workflows facilitate scientists' job in the composition of their scientific applications as a set of computational tasks connected by data- and control-flow dependencies that can be executed in parallel on distributed resources, such as clouds [20], to reduce the time required to obtain scientific results. Although both business and scientific workflows share basic concepts, both have specific requirements and hence need separate consideration. This thesis, however, focus on scientific workflows, and from now on, we will refer to them simply as *workflows*.

Since directed acyclic graphs (DAGs) are commonly used by the scientific and research community to express scientific applications as scientific workflows, the scope of this thesis is limited to workflows modelled as DAGs, which by definition have no cycles or conditional dependencies. Formally, a DAG $\mathcal{G} = \{\mathcal{U}, \mathcal{E}\}$ representing a scientific workflow application is composed by a set of tasks $\mathcal{U}$ and a set of edges $\mathcal{E}$. Each node $u_i \in \mathcal{U}$ represents a computational tasks of the workflow while each edge $e_{i,j} \in \mathcal{E}$ represents a data dependency between u_i and u_j, i.e., the data produced by u_i that will consumed by u_j. On the basis of this definition, a task u_j can not be executed until all its predecessors have completed their executions and sent their output data file to the corresponding computer r where u_j is scheduled to run. Once all u_j's input data are available in r, u_j starts its execution.

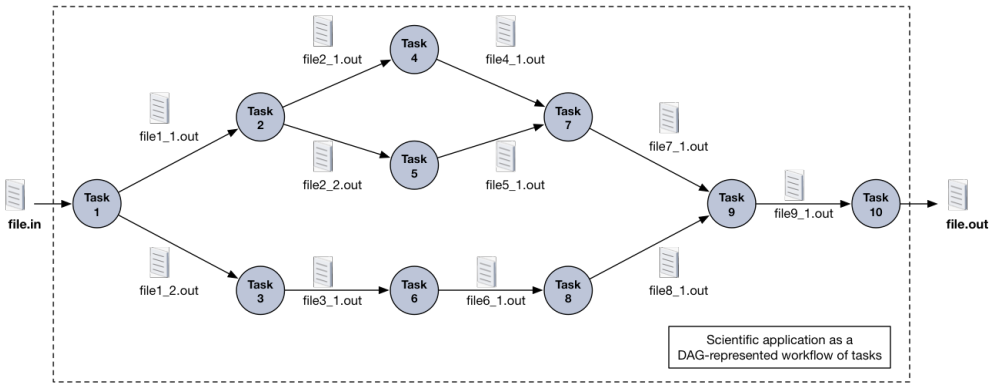
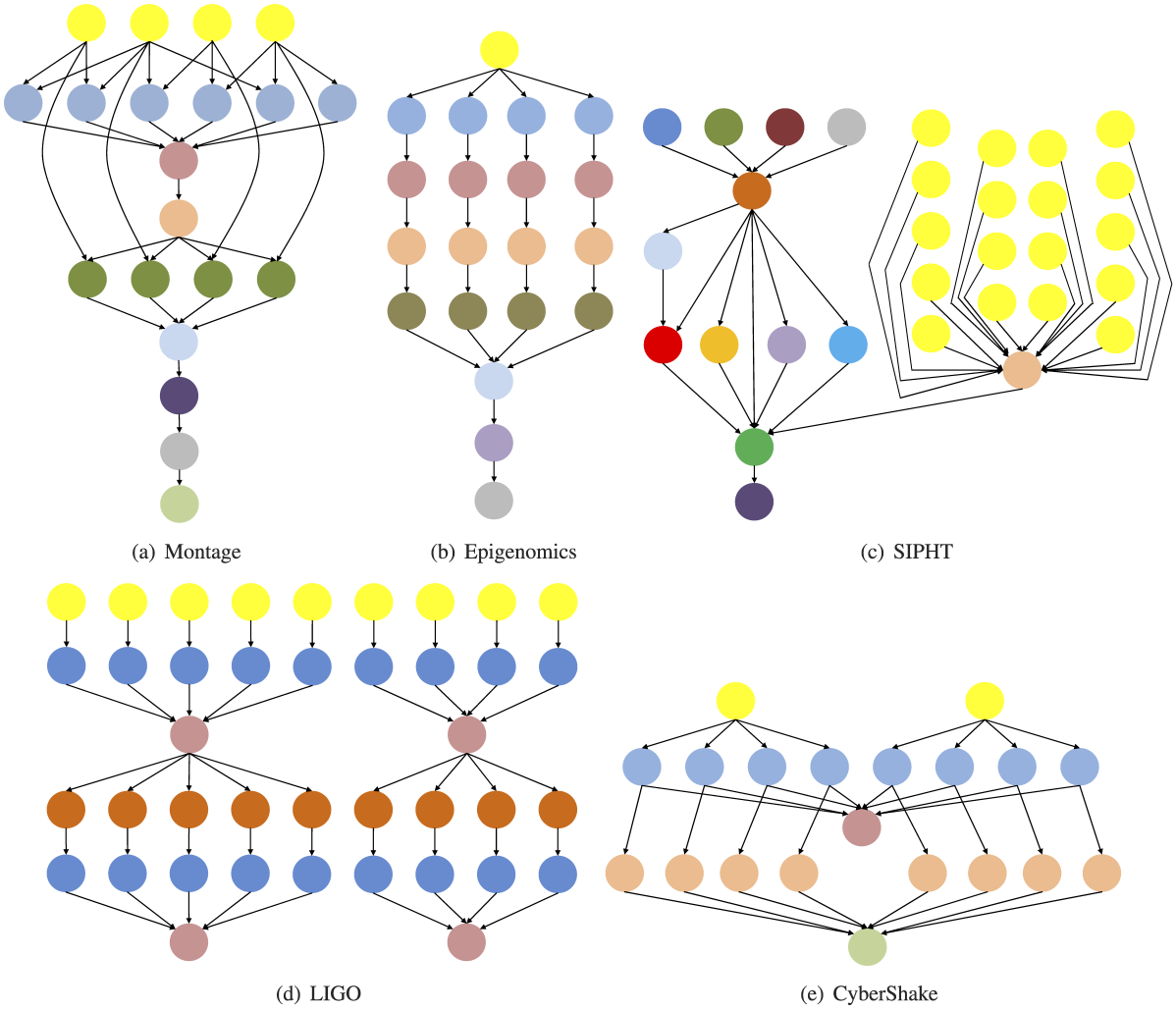


Figure 2.2: A sample workflow with 10 tasks and 12 dependencies. The DAG nodes represent computational tasks and the DAG edges the data dependencies between these tasks. Since tasks are loosely coupled, for example, through files, they can be executed on geographically distributed clusters, clouds, and grids, or multiple computational facilities and/or scientific instruments at user facilities.

A workflow is expressed, therefore, as a set of computational tasks with dependencies between them. In scientific application, it is common for the dependencies to represent a data flow among tasks through files; the output data file generated by one computational task becomes the input data file for the next dependent one. Figure 2.2 shows a sample workflow with 12 tasks and 12 dependencies. In practice, scientific workflows comprise therefore loosely-coupled parallel tasks connected by data- and control-flow dependencies represented by data files [85]. The terms *workflow* and *DAG*, within this thesis, refer to the scientific application composed of dependent tasks and are used interchangeably.

Many scientific areas have embraced workflows as a mean to express complex computational problems that can be efficiently processed in distributed environments such as cloud computing [20, 54, 85, 86, 88]. Figure 2.3 shows the structure of five scientific workflows, and their full characterisation is presented by Juve et al. in [87]. For example, the application Montage [119] is an astronomy toolkit for constructing science-grade mosaics in the flexible image transport system (FITS) format, the international image format standards used in astronomy [20]. It is characterised by being I/O intensive that is used to create custom mosaics of the sky on the basis of a set of input astronomical images. It enables astronomers to generate a composite image-mosaics of a region of the sky that is too large to be produced by telescopes. The size of the workflow depends on the squared degree size of the region of the sky to be generated [87]. For example, for a 1-degree square mosaic of the sky (the moon is 0.5 degrees square), a workflow with 232 tasks is executed. For 10-degree square mosaic, a 20,652 tasks workflow is executed, dealing with an amount of data near to 100GB. The full sky is around 400,000 square degrees [57]. During the workflow execution [118], the geometry of the output image is calculated from that of the input images. Afterwards, the input data is reprojected so that they have the same spatial scale and rotation. This is followed by a standardization of the background of all images. Finally all the processed input images are merged to



(a) Montage (b) Epigenomics (c) SIPHT

(d) LIGO (e) CyberShake

Figure 2.3: Structure of five realistic scientific workflows: Montage (Astronomy), Epigenomics (Bioinformatics), SIPHT (Bioinformatics), LIGO (astrophysics), and Cybershake (Earthquake science). For purposes of illustration, these structures are represented in their reduced form. Figure adapted from [1] and [87].

create the final mosaic of the sky region. An example Montage workflow that consists of 10,429 tasks, requires 4.93 CPU-hours to execute, reads 146.01 GB of input data, and writes 49.93 GB of output data [87].

Epigenomic is an example of scientific workflow application in bioinformatic research field. It maps epigenomic state (short DNA segments) of human cells on a genome-wide scale [88]. The workflow splits several input segment files into small chunks, reformats and converts the chunks, maps the chunks to the reference genome, merges the mapped sequences into a single output map, and computes the sequence density for each location of interest in the reference genome [85]. For mapping the human DNA sequences to a reference chromosome 21, the workflow consists of 529 tasks, reads 24.14 GB of input data, and produces 5.36 GB of output data [87]. The Epigenomics workflow is a CPU-bound application that automates the execution of various genome sequencing operations. It spends 99% of its runtime in the CPU and only 1% on I/O and other activities [89]. Another bioinformatic application is the SIPHT, a project that uses a workflow to au-

tomate the process of searching for small RNA encoding genes (sRNAs) that regulate processes such as secretion and virulence in bacteria. The sRNA Identification Protocol using High-throughput Technology (SIPHT) application [95] uses a workflow to automate the search for sRNA encoding-genes for all bacterial replicons in the National Center for Biotechnology Information (NCBI) database. On average, a typical SIPHT run requires approximately 1.23 CPU-hours to execute 31 tasks, and transfers less than 2 gigabytes of read and write data [87].

The Laser Interferometer Gravitational Wave Observatory (LIGO) [55] is a memory intensive application used in the physics field with the aim of detecting gravitational waves through a network of gravitational-wave detector, with observatories in Livingston, LA and Hanford, WA. The observatories' mission is to detect and measure gravitational waves predicted by general relativity – Einstein's theory of gravity – in which gravity is described as due to the curvature of the fabric of time and space. Gravitational waves interact extremely weakly with matter, and the measurable effects produced in terrestrial instruments by their passage will be minuscule. In order to increase the probability of detection, a large amount of data needs to be acquired and analysed which contains the strain signal that measures the passage of gravitational waves. LIGO applications often require on the order of a terabyte of data to produce meaningful results. The LIGO workflow is very complex and is composed of several sub-workflows. A typical LIGO contains 25 sub-workflows with more than 2000 total tasks. Other LIGO instances contain up to 100 sub-workflows and 200,000 tasks [88]. An interesting aspect of the LIGO is that it reads a large amount of data (213 GB), but writes very little (around 50 MB) when compared with previously workflows that generate significant output data.

An example in earthquake science is the CyberShake [87], a I/O- and memory-intensive application used by the Southern California Earthquake Centre to generate seismic hazard maps. These maps indicate the maximum amount of shaking expected at a particular geographic location over a certain period of time and are used by civil engineers to determine building design tolerances. The CyberShake workflow begins by generating strain green tensors for a region of interest via a simulation. These strain green tensor data are then used to generate synthetic seismograms for each predicted rupture followed by the creation of a probabilistic hazard curves for the given region. The CyberShake workflow execution requires significant amounts of large high-performance computing resources connected via network links with high bandwidth [42, 52]. On average, a typical CyberShake run requires approximately 9,192.45 CPU-hours to execute 815,823 tasks, reads 217 TB of input data, and writes 920 GB of output data [87].

As it can be noted, the workflow is normally classified into three different categories depending on the nature of its computational tasks [88]: CPU, memory, or I/O intensive (or a combination of these). CPU-intensive applications are composed of tasks that spend most of their time performing CPU operations, like the Epigenomics and SIPHT applications, whereas memory-bound applications are the ones that most of the tasks require a high physical memory usage during their executions; for instance, we have the CyberShake and LIGO applications that fit in this category. Lastly, I/O-bound applications are represented by tasks that require and produce large amounts of data and hence spend most of their time performing I/O operations, and for this case, we have shown the

Montage and CyberShake as good examples of I/O-bound applications.

Indeed, scientific workflow systems have become a necessary tool for many applications, enabling the composition and execution of complex analysis on distributed resources [12, 53, 129]. Along with this, scientists must configure these resources (virtual machines in case of using cloud computing) to suit the hardware and software need of a specific workflow application, and with the assistance of scheduling algorithms (workflow schedulers), they can efficiently run their applications to obtain results in a reasonable amount of time [19, 32]. In fact, in recent years, by providing resources on a simple, cost-effective way, cloud computing is revolutionising the way how e-Science is being done [20, 54, 56]. According to the type of a workflow application, for instance, scientists can quickly lease faster-processing virtual machines, or those ones with high-speed memory or hard disk, to satisfy the applications' computational demands. In this way, scientists should also address financial issues when using cloud resources because they are charged on a pay-per-use basis. Since application executions are likely to occur on a tight budget, scientists also must take into account the price of clouds resources to make these executions cost-effective. They need to find a trade-off between performance and cost to avoid paying unnecessary and potentially prohibitive prices in the rental of resources to carry out science experiments cheaply. Therefore, the pricing model of resources is another challenge that must be addressed by scientists when employing a cloud-based environment system to run e-Science applications. In fact, as the next chapter shows, cost-aware schedulers can assist scientists to tackle this financial trade-off.

Therefore, several scheduling challenges arise from using on-demand, elastic, pay-as-you-go resource model offered by cloud computing to execute scientific workflows. When compared to other distributed systems such as grids, clouds offer more control over the type and quantity of resources used [118]. This flexibility and abundance of cloud resources creates the need for a strategy of resource provisioning that works together with the scheduler; a scheduling algorithm that decides the type, price, and number of resources to use and when to lease and to release them. The next section focuses on describing the workflow scheduler, as well as an overview of the workflow scheduling problem.

2.3 An Overview of the Workflow Scheduling Problem

On one hand, a scientific workflow can be seen as a high-level specification of a set of computational tasks and the dependencies between them that must be satisfied in order to accomplish a specific scientific results. For instance, a data analysis protocol of a typical scientific applications modelled as workflows often consists of a sequence of pre-processing, analysis, simulation, and post-processing steps. On the other hand, a workflow is a computer program and it can be expressed in any modern programming language, such as Java or Python. However, the task of writing a computer program to schedule a set of tasks on a wide-area distributed resources goes well beyond the programming skills or patience of most scientific users. The aim of a scheduler is to provide scheduling decisions to simplify the programming effort required by scientists to proceed with the scheduling of the tasks of a science experiment to the underlying distributed resources.

The study of scheduling became important in the early twentieth century, with Henry Laurence Gantt being one of the leading pioneers in this research field [32]. Initially used in industrial production, the study of scheduling evolved over the years and currently is widely used in computational environments, and is now a focus of research. Scheduling is a generic problem that is found in the most varied areas, being defined in different ways according to the problem addressed. Scheduling, as a decision-making process, plays an important role in most manufacturing and production systems as well as in most information processing environments. The definition given by Pinedo in [112] is as follows:

> *Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimise one or more objectives.*

In fact, the resources and tasks in a scheduling problem can take many different forms. For example, the resources may be employees in a company, runways at an airport, or even processing units in a computing environment system, and so on. Regarding the tasks, it may be operators in a production process, take-offs and landings at an airport, and executions of computer programs, and so on. Also, each task may have a certain priority level and an earliest possible starting (or finishing) time. The objectives of a decision-making process can also take many different forms. One of the objectives may be the completion time of the last task, which must be equal to or less than a specified deadline, or it can be the minimisation of the overall monetary costs involved in the decision-making process or the maximisation of the use of available resources. In this thesis, the tasks are a set of dependent computational tasks of a scientific application modelled as a non-cyclic workflows (DAG) and the resources are a set of distributed heterogeneous virtual machines (VMs) within a public or hybrid cloud.

The scheduling problem is usually referred to in the three-field notation $\alpha \mid \beta \mid \gamma$ [112]. The α field describes the processing environment system and contains just one entry. The β field provides details of processing characteristics and constraints and may contain no entry at all, a single entry, or multiple entries. Lastly, the γ field describes the objective to be minimised (or maximised) and often contains a single entry. According to the Pinedo's notations, this thesis focuses on the scheduling problem in which the processing environment system is composed of m unrelated machines in parallel ($\alpha = R_m$), where there are constraints of precedence between tasks ($\beta = prec$). The objective can assume several forms in this thesis, but in most case we attempt to minimise the *makespan* (completion time) of the workflow ($\gamma = C_{max}$). Thus, the general problem we have addressed is referred to $R_m \mid prec \mid C_{max}$. Whenever we only want to specify the target system and tasks characteristics, but not a particular objective function, the three-field notation of the scheduling problem can be reduced to $R_m \mid prec \mid *$.

In fact, when all the parameters of the system and of the tasks are known in advance, the scheduling algorithm is referred to as a *deterministic* scheduling model [93, 112]. In deterministic scheduling, therefore, it is assumed that all problem data is known with certainty in advance (Figure 2.4). The solution of such a problem is a schedule – a set of start execution times for all the tasks. However, in some cases, parts of the input data

may be subject to large fluctuations and are not known with certainty in advance. In this case, the input data can be *given* as (distributions of) random (stochastic) variables, and the scheduler that natively deals with it employs probability techniques, which end up classified as a *stochastic* scheduling model [93]. Due to the stochastic nature of the input data, their solution is not a schedule (like in deterministic models), but rather a prescription that tells how to generate a schedule, based on the past evolution of the problem [128]. Stochastic scheduling problems as well as the employed techniques for their solution may contain elements from combinatorial optimisation, stochastic dynamic programming, and also probability theory [112]. For stochastic schedulers, it is important to note that input information can not be confused with random values. Probability theory are useful for modelling scheduling algorithms in which there is a certain regularity in the occurrence of events [19]. This regularity allows events to be described by probability distributions given as input data to the scheduler. In the case of lack of this regularity (distributions), which makes it difficult to use stochastic tools to modelling algorithms, there are studies that employ the fuzzy theory to develop schedulers [19]. Fuzziness may not be confused with probability. A statement is probabilistic if it expresses a likelihood or degree of certainty if it is the outcome of clearly defined but random occurring events. Fuzzy schedulers are used when there are no precise criteria for defining the scheduler's input data [114, 115]. Notice that the terminology *deterministic scheduling* is generally used to make a distinction to *stochastic* or *fuzzy* scheduling models. Unless explicitly stated from now on, the term *scheduler* is used in this thesis to refer *deterministic schedulers* because we focus in this type of scheduling model.

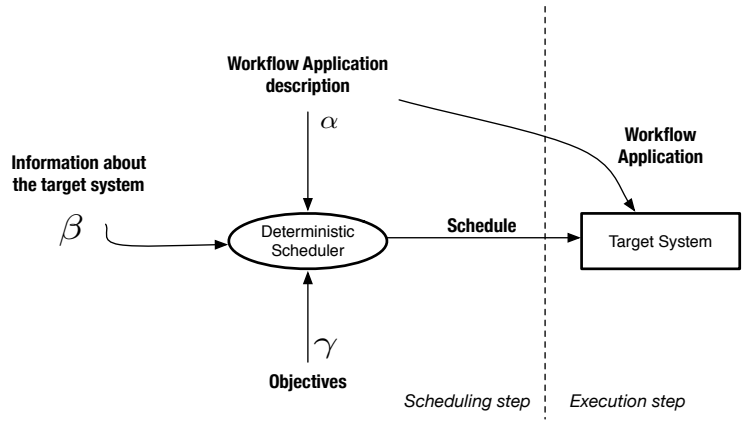


Figure 2.4: Illustration of the $\alpha \mid \beta \mid \gamma$-notation of a deterministic scheduler.

Scheduling is fundamental to achieve good performance in the execution of application workflows in distributed systems, and in a heterogeneous system like clouds, new variables must be considered. Thus, there has recently been a growing interest in algorithms for scheduling workflows in heterogeneous resources [56]. Part of the motivation of these works comes from the diffusion of heterogeneous processing platforms, such as clouds, and the growing interest in applications that can be modelled by DAGs, such as scientific workflows [63, 85, 86, 129]. To optimise the desired objective function, the scheduler can use information about the current state of the system, which guides the decision-making process about the resource in which each task is executed. The input of the scheduling

algorithm must include the DAG that represents the workflow of tasks and their dependencies, as well as information about the target system, including the processing capacity of each resource and the available capacities of the network links.

In the scheduling of a workflow in a cloud-based distributed system, the scheduler is the element responsible for determining which task of a workflow will run on what virtual machines, distributing the workflow's tasks to different processing elements (VMs) so that they can run in parallel, speeding up the execution of the application. Thus, the execution time and monetary cost of running a set of computational tasks on a pay-per-use basis distributed system are strongly dependent on the scheduler decisions. In fact, the scheduler performs the decision-making process according to an objective function, which can have one or more criteria to be optimised, such as the schedule length (makespan), communication costs, resource utilisation, fairness, and so forth.

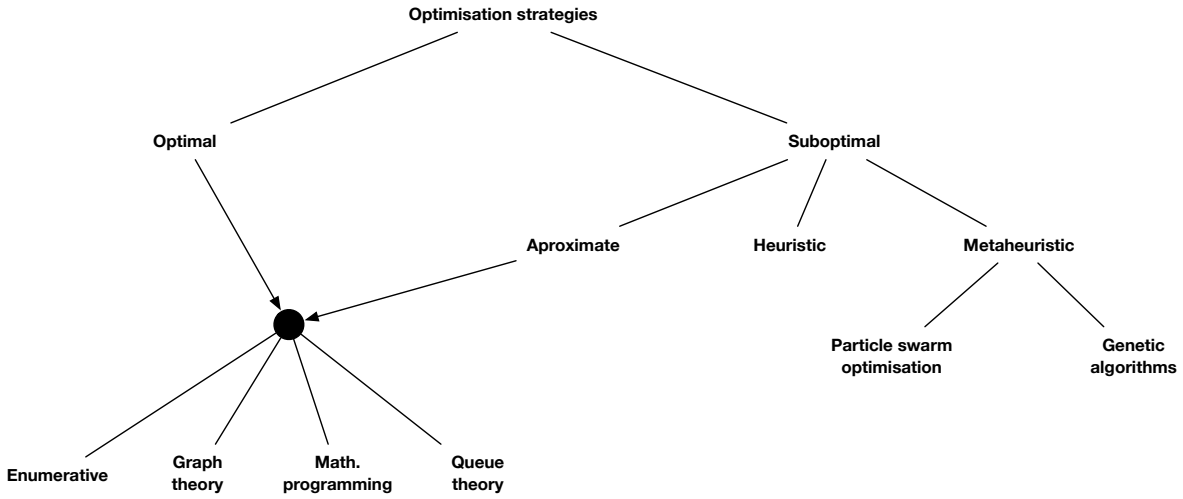


Figure 2.5: Types of optimisation strategies (based on [33, 45, 118])

The scheduling problem involved is known to be NP-complete in general [112], including the scheduling of workflows in heterogeneous environment systems discussed in this thesis. Consequently, its optimisation version is NP-Hard and thus there is no known algorithm which finds the optimal solution in polynomial time [105, 112]. According to the definition of Casavant and Kuhl in [45], scheduling algorithms can be classified as optimal or suboptimal. Due to its NP-completeness, finding *optimal* solutions is computationally expensive even for small-scale versions of the problem, rendering this strategy impractical in most real-scenario situations [118]. There are several methods that can be used to find optimal schedule solutions [45]. In particular, as Figure 2.5 shows, four strategies are identified in [45]: solution space enumeration and search, graph theoretic, mathematical programming, and queueing theoretic. For the workflow scheduling program, most relevant algorithms are those based on solution space enumeration and mathematical models. For example, in the surveyed algorithms, integer linear program (ILP) is a special case of mathematical programming that has been used as a tool to modelling scheduling algorithms to obtain optimal schedule solutions [16, 19, 69, 71, 72]. However, the overhead of finding the optimal solution for large-scale workflows through ILP-based schedulers may not be an attractive strategy in most cases since these schedulers are expected to run for

long periods to find still suboptimal solutions [72]. Even for the scheduling of small-scale workflows, they may not achieve optimal schedules in a timely manner [72].

Therefore, given the NP-Completeness of the scheduling problem, most algorithms in the literature focus on generating approximate or near-optimal solutions. Significant effort has been made towards the development of algorithms that can provide efficient suboptimal schedule solutions with polynomial time complexity. Therefore, for the *suboptimal* category, as Figure 2.5 shows, we identify three main different methods to develop polynomial-time scheduling algorithms [33, 45, 118]. These techniques and their characteristics are listed below.

- *Approximate*: Approximate algorithms use formal computational models, but instead of searching the entire solution space for an optimal solution, they are satisfied when a solution that is sufficiently "good" is found. In the case where a metric is available for evaluating a solution, this technique can be used to decrease the time taken to find an acceptable schedule. In fact, it returns a solution that is probably close to optimal. It is hard to obtain low complexity approximation algorithms for general problems, but they provide guarantee related to the solution quality, giving bounds from the optimal solution. More general the problem, harder to obtain a satisfactory approximation. Some works that use approximation algorithms to solve the workflow scheduling problem are cited in [64, 109];

- *Heuristics*: Heuristic-based algorithms are usually adapted to the problem at hand and take full advantage of the particularities of this problem. However, as they are often too greedy, they usually get trapped in a local optimum and end up failing to get a optimal solution. In short, heuristic-based algorithms have low complexity and fast execution but offer no guarantee related to the solution quality, providing, therefore, "good" solutions in a "reasonable" amount of time. Examples of scheduling algorithms based on heuristics are the works cited in [31–33, 35, 120]; and

- *Metaheuristics*: It is a high-level problem-independent algorithmic tool that provides a set of guidelines or strategies to programmers to develop algorithms that can explore the solution search space efficiently in order to find near-optimal solutions. As metaheuristics-based algorithms are not greedy like heuristic-based ones, the execution time to achieve good-quality solutions is usually higher than that of heuristics. At the expense of exploring more thoroughly the search space to hopefully get a better solution (that sometimes will coincide with a global optimal), a deterioration in its runtime may be acceptable. This is represented on the stop condition (usually the number of iterations) imposed by the programmer that indicates the thoroughness of the algorithm in the analysis of the search space to find a good, perhaps optimal, solution. However, like to heuristics, algorithms based on metaheuristics also provide no guarantee related to the solution quality; local optima is frequently stated as the final solution. In short, when compared to heuristic-based algorithms, metaheuristic approaches are generally more computationally intensive and take longer to run; however, they also tend to find more desirable schedules as they explore different solutions using a guided search. Particle swarm optimisation [90] and genetic

algorithms [148] are two examples of metaheristic tool used in the development of workflow scheduling algorithms. Examples of workflow scheduling algorithms based on metaheuristics are the works cited in [65, 74, 75, 103, 104, 143, 148].

The most commonly used approach in the workflow scheduling literature is the use of heuristics and meta-heuristics, which combine good quality solutions in general and low complexity as well. As a consequence of these characteristics, the algorithms we develop presented in this thesis are based on the use of heuristics and metaheuristics.

Scheduling algorithms can be classified as schedulers for *independent tasks* and schedulers for *dependent tasks* [32, 33]. Seen as a bag-of-tasks, the scheduling of independent tasks does not need to tackle with tasks communication costs and dependencies, thus each task can be scheduled independently. However, the focus of this thesis is the scheduling of dependent tasks, since scientific workflows are composed of tasks with dependencies. Therefore, in this type of scheduling, the algorithm must take into consideration the communication costs proceeding from data dependencies between tasks. These costs are implicit when tasks are sent to execution over a set of distributed resources, such as clouds, with the heterogeneity of network links with uncertain available bandwidth estimates playing an important and challenge role in this aspect.

Scheduling algorithms can also be classified as *static* schedulers and *dynamic* schedulers [32, 33]. Static schedulers schedule all tasks using the information initially available about the resources of the target system, while dynamic schedulers may schedule tasks in turns, updating the information about resources at each scheduling turn. It is important to note that static scheduling algorithms are useful to the development of dynamic schedulers, which can use the static scheduler to schedule parts of the workflow in turns. In that sense, we consider that a good static scheduler is the starting point for a good dynamic scheduler. In this this thesis we deal with static schedulers for applications modelled as a set of dependent tasks represented by non-cyclic workflows as DAGs.

2.4 Final Remarks

This chapter presented background and key concepts needed to understand the content of this thesis. The definition of the cloud computing was presented, especially that one regarding hybrid clouds which is the main focus of this thesis. Scientific applications modelled as a workflow of tasks were also presented, including its characteristic of containing loosely-coupled tasks stitched by data- and control-flow dependencies through files. The main characteristics of a deterministic workflow scheduler were introduced as well, including the assumption that all input data is known with certainty in advance.

With respect to the practical implication of deterministic models of workflow scheduling can be criticised. The reason is obvious: In many practical situations we are faced with uncertain information about the input data. Actually, one of the main issues faced in deterministic scheduling models is the quality of the information that is provided as input data to the scheduler to produce a schedule. Especially in hybrid clouds, this can concern with the information about the available bandwidth between clouds. Resources in hybrid clouds, consisting of private and public resources, are normally connected through

Internet links whose available bandwidth fluctuates considerably because of the shared communication infrastructure. In fact, as it is discussed in the next chapter, bandwidth estimations will be given by monitoring/measurements network tools in ranges rather than deterministic values [14]. It also will be discussed that the use of imprecise bandwidth estimates may cause a deterministic scheduling algorithm (scheduler) designed especially for hybrid clouds to generate schedules that result in slow execution time and/or poor application execution performance. In general, deterministic schedulers for hybrid clouds typically ignore the uncertainties of available bandwidth, making them prone to produce schedules that are *sensitive* (non-robust) to these uncertainties. On a contractionary basis, these schedulers are especially designed to work in an environment where bandwidth estimates are very likely to vary and cannot be estimated accurately in advance.

In this thesis a schedule is said to be *robust* if it is able to absorb some degree of uncertainty in bandwidth estimates. Robust schedules are extremely needed, for instance, in mission-critical and time-critical (deadline constraints) application executions. Robust workflow scheduling algorithms that are able to identify these aspects can provide a robust schedule that is *insensitive* to these uncertainties, tolerating variations in bandwidth estimates up to a certain degree of uncertainty. In order *not* to discard these existing non-robust deterministic workflow schedulers and provide robustness to them, Chapter 3 proposes a mechanism that attempts to "robustify" these schedulers in the face of uncertainties naturally embedded in bandwidth estimates that are furnished by monitoring/measurement networking tools.

Chapter 3

A Procedure to Provide Robustness for Sensitive Workflow Schedulers to Uncertainties of Available Bandwidth

A schedule is said to be *robust* if it is able to absorb (or be insensitive to) the (natural) uncertainty in the input data to guarantee that the workflow execution will experience a low or no degradation in its performance. This chapter proposes a proactive procedure to provide a certain level of robustness to non-robust schedulers so that they can be able to produce robust schedules under the uncertainties of available bandwidth. As bandwidth is a shared resource in networks, it is very likely that it has a high variability in its availability, which makes network tools unable to accurately estimate in advance [14]. As the bandwidth estimates stemming from using these tools are not accurate information, deterministic schedulers end up producing non-robust (sensitive) schedulers under these uncertainties. Therefore, this chapter focuses on the use of uncertain bandwidth information by deterministic schedulers (that were not designed to cope with such inaccuracies) and try to make them to produce schedules with a certain level of robustness under imprecise bandwidth estimates by employing the proposed procedure. The procedure does not change the scheduling algorithm or the scheduler itself, but is a procedure that assists the scheduler in producing robust schedules under imprecise bandwidth estimations. The procedure deflates the estimated available bandwidth before furnishing it to the scheduler. Instead of using imprecise bandwidth measurements directly provided by monitoring networking tools, deterministic schedulers will use a deflated value of the original measurements in an attempt to *proactively* mitigate the production of poor (non-robust) schedules. The deflating factor is computed by the procedure through a multiple linear regression using a historical information of previous workflow executions.

This chapter is organised as follows. Section 3.1 introduces the scenario where the proposed procedure was developed, while Section 3.3 presents related work. Section 3.4 reviews some deterministic workflow schedulers specialised for hybrid clouds that were not designed to handle inaccurate information about the available bandwidth, making them sensitive to the uncertainties of available bandwidth. The proposed procedure is introduced in Section 3.5 and evaluated in Section 3.6. The evaluation results are discussed in Section 3.7. Moreover, a performance comparison between a robust-designed scheduler

and its non-robust version but using the proposed procedure is discussed in Section 3.8, while Section 3.9 provides final remarks and concludes this chapter.

3.1 Hybrid Cloud Scenario

In public clouds, users are charged on a pay-per-use basis, and therefore, it is preferable to first use the available owned free of charge infrastructure (private cloud) before starting leasing resources from public cloud providers when needed. The composition of resources derived from private and public clouds is called a hybrid cloud. To make such composition, several relevant questions need to be answered for applications modelled as workflow to achieve high performance executions: *how many and which resources should be leased to execute an application? Which application (or part of) should be executed on leased resources in order to improve performance, reduce costs, and met deadlines?* These questions are answered by a schedule produced by a workflow scheduler, which makes decisions on resource allocation based on criteria such as execution time and leasing cost.

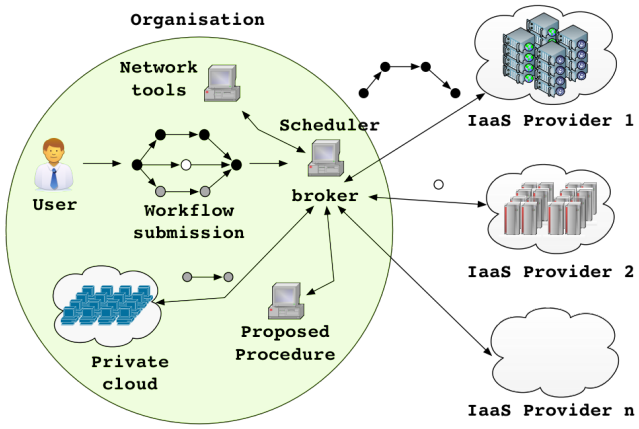


Figure 3.1: Hybrid cloud infrastructure and the proposed procedure.

Figure 3.1 illustrates a hybrid cloud scenario composed of the resources of the private cloud as well as those of one or more public cloud providers. The role of the scheduler is to decide which resources should be leased from the public clouds to guarantee, for instance, the execution time of the workflow be within a specified maximum execution time (deadline) or the execution cost be within a specified budget. In this scenario, after the workflow has been submitted by the user, a broker calls the scheduler to start the decision-making process for the workflow execution. Besides deciding which resources will be used to execute the workflow, the scheduler also determines which part of the workflow will run in each resource of each cloud provider. Therefore, we assume an organisation with a broker in its premises which is responsible for renting the resources from public cloud providers to be connected with those of the private cloud to compose the so-called hybrid cloud. The renting process is carried out according to the information extracted

from the schedule produced by the scheduler in order to run the workflow according to this schedule.

Therefore, the broker receives requests for workflow executions and rents resources from the public cloud based on the produced schedules (scheduler's decision). However, in our scenario, before executing the scheduler immediately as soon as the workflow is sent for execution, the broker is responsible for invoking the proposed procedure in order to make the deterministic schedulers produce schedules with a certain level of robustness under uncertainties of available bandwidth in intra- and inter-cloud links clouds. After that, the broker rents the necessary resources according to the information given by the robust schedule.

3.2 Impact of Using Imprecise Bandwidth Information on Workflow Scheduling

The input data to a deterministic scheduler must include information about the DAG representing the workflow of tasks and their dependencies, as well as information about the target system, including the processing capacity of each resource and the available capacities of the network links. Information about the workflow can be gathered from the submitted DAG, while information about the target system is gathered from a resources information repository that can be fed by resource measurement/monitoring tools. Therefore, having all this information in advance, a deterministic scheduler is capable of producing a schedule for the workflow and estimating its makespan and execution costs, besides informing whether the execution will or will not meet a stablished deadline.

Figure 3.2 illustrates the inputs necessary for a deterministic scheduler to produce a schedule and gives an example of a hypothetical schedule. The DAG contains information on the computational requirements of its tasks as well as information on the amount of bytes to be transmitted to resolve each data dependency among tasks. Nodes in the DAG are labeled with their computation cost, millions of instructions (MI) for instance, while edges are labeled with their communication cost, bytes to transmit, for instance. MI per second (MI/s) represents resource processing capacities given by resources' owner (or provider), while megabits per second (Mbps) indicates link bandwidths furnished by monitoring networking tools. The scheduler combines all of this information to compute how long each task takes to run on each resource, how long each data transmission would take according to the resource assigned to each task, and how much a given task assignment would cost. All this to find a schedule for the workflow that satisfies the objective function of the scheduling problem.

Figure 3.3 depicts an example of a schedule of a workflow application by using the Hybrid Cloud Optimised Cost (HCOC) scheduler [35] (detailed in Section 3.4.3). HCOC is a deterministic deadline- and cost-aware scheduler that decides which resources should be leased from the public cloud to join the private cloud in order to provide sufficient processing power to execute the workflow cost-effectively within a given execution time (deadline). As illustrated in Figure 3.3, a schedule usually contains details concerning in which resource each task of the workflow will run, as well as information regarding the

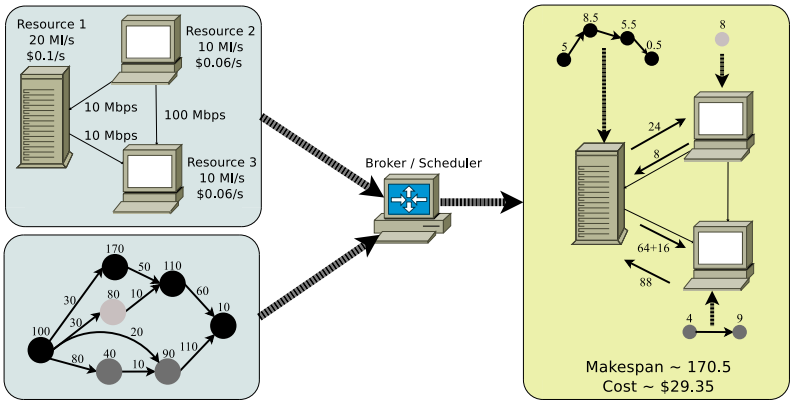
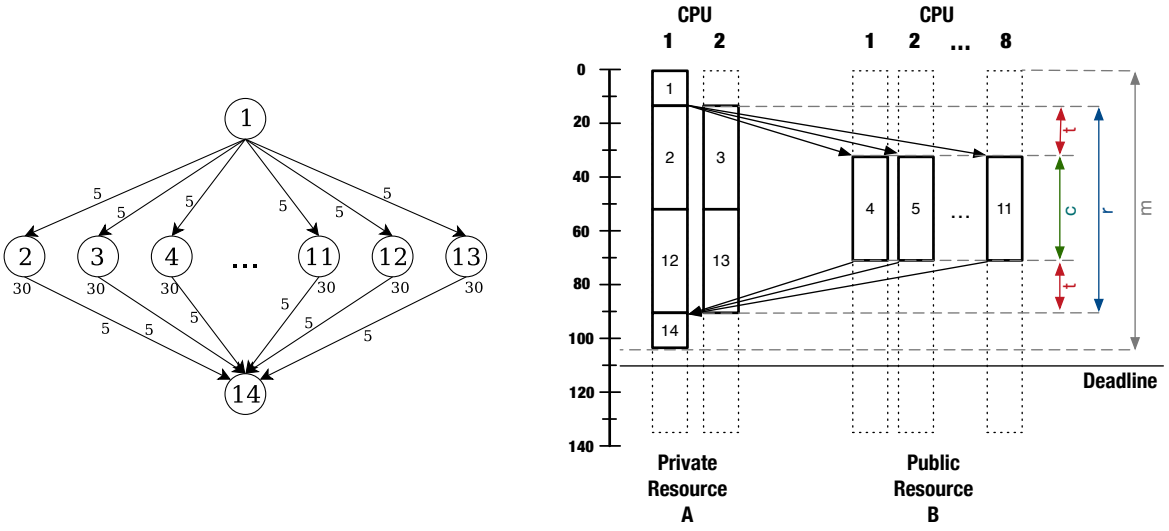


Figure 3.2: Information about the DAG and the target system used as input to the scheduler (adapted from [34]).

start and finish time for those executions. Therefore, a schedule summarises estimations for the workflow execution time (makespan) as well as the costs involved with resource rentals for this execution. Figure 3.3 illustrates an example in which the HCOC decided to rent a 8-core virtual machine in order to satisfy a specified deadline for the execution of a workflow represented by a 14-nodes fork-join DAG. In this illustrative example, the HCOC decided to rent a resource with 8-core because the private 2-core resource was not able to meet the desired deadline.



(a) Example of a fork-join DAG with 14 nodes and 24 edges [35]. Nodes in the DAG are labeled with their computation cost (number of instructions, for instance), while edges are labeled with their communication cost (bytes to transmit, for instance).

(b) Example of a schedule, which states the location and time of each task execution; m is the estimated makespan, r the estimated renting time of B, c the estimated computation time of B, and t the estimated data transfer time between A and B. Estimates given by the schedule assert that the deadline is satisfied

Figure 3.3: Example of a DAG (left) and its schedule (right) under a deadline constraint.

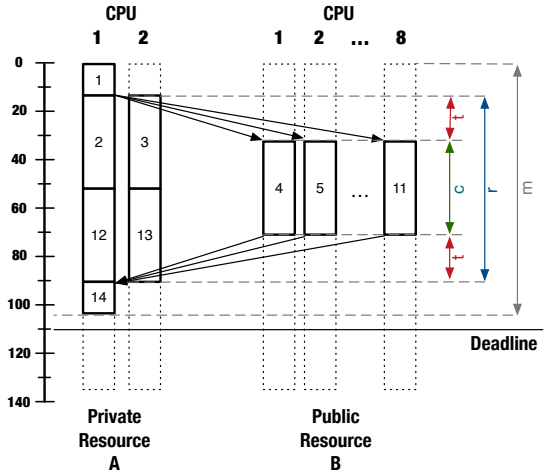
In addition to being available in advance, the input data is usually considered accurate. These data, however, are likely to experience (sometimes high) variability [19]. For example, the uncertainties of application demands arise from the lack of precision in

estimating the computation requirements for task executions and the amount of data to be transferred between these tasks. In these cases, a DAG may not be properly modelled by the users to faithfully represent the workflow application [14, 16, 18]. Another example is the uncertainties of available bandwidth in inter-cloud and intra-cloud links. This is normally related to the nature of measurement and monitoring tools, in which estimations are quite often given in wider ranges rather than as deterministic values [14,82,91,92,108,116,121–123,137]. Also, variabilities regarding the description of the processing capacities of cloud resources may occur by providers as well since the resources are virtualised and share physical resources [3, 81, 85, 102, 108]. Imprecise information (estimations) of both applications demands and resource availability impose additional challenges to workflow scheduling [14, 34]. Although all input data may contain inaccuracies, this thesis focuses only on the performance of schedules that are produced using inaccurate bandwidth estimates in intra-cloud and inter-cloud links of a hybrid cloud.

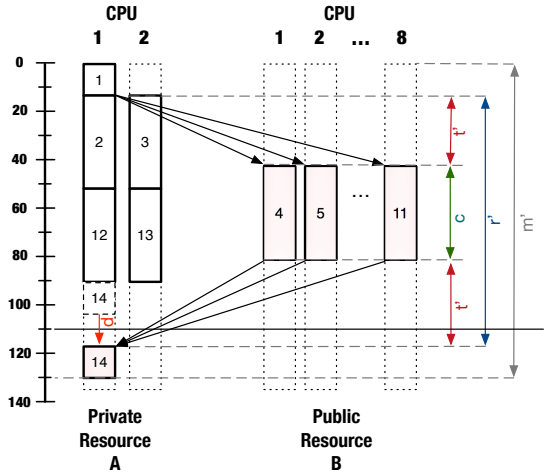
The expected application execution stated by schedules produced by deterministic schedulers using inaccurate input data may be remarkably different after the application has been executed. In this case, the produced schedule is considered poor since the execution of the application does not occur as expected. Scheduling decisions on resource allocation using imprecise information about the available bandwidth can be immensely ineffective [19, 24, 46, 67, 68, 74]. For example, the use of poor schedule may lead the application having seriously delay in its runtime (makespan), which can lead to deadline violations. These delays can also lead to increased costs, as pay-per-use resources may have to stay active longer to wait for the completion of data file transfer. To illustrate an example of how inaccurate available bandwidth can push the scheduler to produce poor schedules, Figure 3.4 shows a comparison between the expected execution of a workflow given by the schedule of Figure 3.3b and a hypothetical snapshot of the workflow execution using this schedule. Because the variabilities on the available bandwidth in inter-cloud links may occur during the execution of the workflow, the total time of data transfers between the resources A and B estimated to take $2t$ took $2t' > 2t$. This bandwidth variability may enlarge the makespan, leading to a potential deadline violation. Also, the rent of B becomes more expensive than expected since the resource remained active for longer than planned ($r' > r$), leading to a potential budget violation as well.

The precision of input data provided to the scheduler, especially those ones regarding the available bandwidth of links that connect resources located in different sites, is indeed essential to the effectiveness of the schedule generated by a deterministic scheduling algorithm [19]. The example illustrated by Figure 3.4 reinforces that deterministic schedulers are prone to produce poor schedules that are sensitive (non-robust) to the uncertainties of available bandwidth. Although these schedulers are not designed to address such uncertainties, they are design to be employed in a environment where bandwidth estimates are very likely to vary and cannot be estimated accurately in advance [14]. This issue can be potentially reduced by adoption of procedure that can assist schedulers in taking into consideration the uncertainties embedded in bandwidth estimates, making them producing insensitive (robust) schedules to these uncertainties.

Figures 3.5a and 3.5b illustrate the contribution of the proposed procedure. It shows the executions of the schedules produced by the HCOC when the procedure was not and



(a) Expected execution given by the schedule of Fig. 3.3b.Schedule produced by assuming bandwidth estimates as accurate information. As HCOC neglects the variability of data transfer time that may occur between resources, it ends up producing sensitive schedules to these variabilities. The schedule given by HCOC clearly expects to meet the deadline.

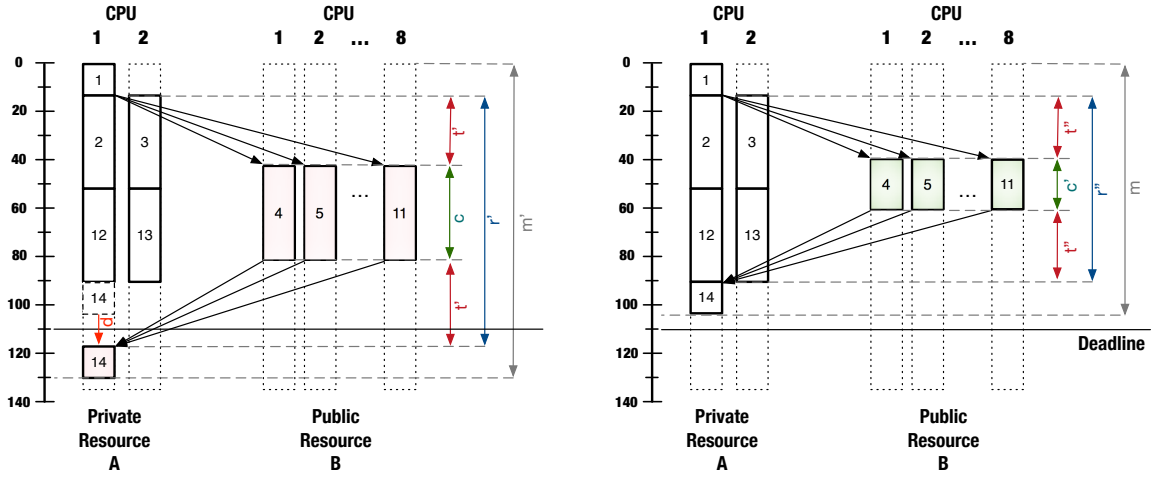


(b) Illustration of a hypothetical execution of the DAG guided by the schedule in (a). Due to the variability of the available bandwidth in inter-cloud links, the estimated data transfer time between A and B took $2t' > 2t$, delaying the start time of the executions of the tasks 4 to 11, and consequently the task 14. The estimated computation time c of B remains the same, but the estimated renting time of B increased to $r' > r$ because $2t' - 2t > 0$, implying an increase in costs of B. The deadline, expected to be met, has missed.

Figure 3.4: Comparison between the expected (left) and a hypothetical (right) of the execution of the DAG of Figure 3.3a.

was used, respectively. By using the proposed procedure, for instance, the HCOC may be able to produce robust schedules that can undergo less degradation in its performance than those produced without using the procedure. In this example, the robust schedule is able to absorb some degree of uncertainty presented in bandwidth estimates. Expecting to have bandwidth variabilities between A and B, the proposed procedure furnishes a deflated value of the measured bandwidth, which makes the HCOC to lease a faster (and consequently expensive) resource to execute the tasks 4 to 11. Expected to have less bandwidth available between A and B, the HCOC's decisions shortens the execution time of those tasks executed in remote clouds, which leaves room to absorb (amortize) some data transfer delays that can occur among the resources. Although the execution might become slightly more expensive when using the proposed procedure, the scheduling objective (i.e., satisfying the deadline in this case) can be easily met regardless the presence of such uncertainties. Note that the example given in Figure 3.5 is just an illustration to show how the proposed procedure can "inject" robustness to schedulers that were not designed to be robust under uncertainties of available bandwidth.

Like HCOC, deterministic schedulers for hybrid clouds presented in the literature do not consider the presence of the uncertainty of available bandwidth in inter-cloud and intra-cloud links. Thus, we propose a proactive procedure that attempts to "*robustify*" these schedulers in the presence of such uncertainties. Note that Figures 3.4 and 3.5 are only illustrations to highlight how the proposed procedure can make non-robust schedulers



(a) Illustration of a hypothetical execution of the DAG when using the schedule of FigURE 3.3b produced by the HCOC scheduler. The HCOC was fed by information about the available bandwidth gathered directly from monitoring/measurement network tools. The deadline, expected to be met, was missed because the schedule was not produced to be insensitive to the variabilities in the available bandwidth.

(b) Illustration of a hypothetical execution of the DAG when using a robust schedule. The HCOC selected a faster resource to shorten the execution time of the tasks 4 to 11 in order to leave some room to absorb some unexpected delays in data transfer between A and B. The deadline was able to be met because the schedule was produced to be robust under variabilities in available bandwidth on channels connecting A and B.

Figure 3.5: An illustration of executions using the schedules produced by the HCOC without using (left) and using (right) the proposed procedure.

to produce robust schedules that are insensitive to the uncertainty of available bandwidth. Sections 3.7 and 3.8 show numerical results that reinforce this argument where the proposed procedure is capable of increasing the robustness of existing schedulers under these uncertainties.

3.3 Related Work

Traditionally, large-scale scientific workflow applications were widely executed on dedicated high performance supercomputer such as private campus clusters or grid computing systems [33, 84]. These systems used to be statically partitioned, were expensive, as well as inefficient in adapting to the surge of demand. The scheduling of scientific workflow applications was, for instance, widely studied on computational grids [15–17, 19, 25, 28, 33, 64, 77, 120, 141, 149]. However, with the emergence of cloud computing, the scientific community regards cloud computing as a potentially attractive source of low-cost computing resources [102]. Instead of using a dedicated supercomputer, which typically offers best-effort quality of service, clouds give more flexibility in creating a controlled and managed computing environment [100]. By providing resources (in terms of virtual machines) on demand as services, cloud computing environments facilitate the running of scientific workflows since scientists can lease resources at anytime on a pay-per-use basis [107]. Due to this, the problem of scheduling workflows has gradually been moved entirely to the clouds [10,21,31,62,69,72,75,94,98,104,106,148]. However, another works,

including this thesis, focus on the scheduling of scientific workflows in hybrid clouds, where part of the workflow is executed in the public cloud, while the remainder is executed in the private (or a community [147]) cloud [20,24,26,27,34,35,46,54,67,68,70,78,86,88,135,150].

Resources in hybrid clouds, consisting of private and public resources, are normally connected through Internet links whose available bandwidth fluctuates considerably because of the shared communication infrastructure [24, 35]. In hybrid clouds, tasks can be allocated to be executed on either a private cloud or a public cloud. The use of imprecise bandwidth estimates may cause a scheduler to generate schedules that result in slow execution time and/or poor application performance [24, 68]. In other words, the capacity of the communication channels connecting these two types of resources impacts the application execution time (makespan) and the cost of the application execution.

This section focuses in showing works related in scheduling workflows under imprecise information about the available bandwidth in intra- and inter-cloud links. Section 3.3.1 describes the impact of using imprecise bandwidth estimates by schedulers that were not designed to use inaccurate information, and presents existing robust schedulers that were designed to address such uncertainties in the production of schedules. Section 3.3.2 highlights that the uncertainties of available bandwidth can be a result of imprecise measurement and monitoring network tools and/or their incapacity of estimating the real (deterministic) value of the available bandwidth during the application scheduling time.

3.3.1 Impact of Uncertainty of the Available Bandwidth on Workflow Scheduling

Regardless the infrastructure where the workflow application is scheduled, grids or clouds, one of the main issues faced in scheduling problems is the quality of the information that is provided as input data to the scheduler to produce a schedule. In hybrid clouds, for instance, the available bandwidth in channels connecting processing resources among private and public clouds impacts the makespan and cost of a schedule. As previous section described, the available bandwidth in intra- and inter-cloud links is quite imprecise [14, 91, 92, 108, 116, 121–123, 137]. Uncertainties on the bandwidth availability can cause delays since Internet links, whose available bandwidth fluctuates considerably because of it is a shared communication infrastructure, are used to transmit data among clouds. To avoid misleading makespan and budget estimation in the workflow scheduling, a robust scheduler in the face of uncertainties of input information is highly necessary.

Although data transfers play an important role in the performance of workflows among clouds, most existing research work on scheduling does not adequately address this issue. Workflow scheduling algorithms often assume that data is transferred directly between execution units. This is the case for the well-known Heterogeneous Earliest Finish Time (HEFT) [130] and other similar heuristics that are based on HEFT [29, 32].

To understand how existing workflow schedulers for hybrid clouds behave when inter-cloud available bandwidth is not accurate, Bittencourt et al. evaluated in [24] the impact of uncertainties of bandwidth availability on the performance of two well-known non-robust deadline-driven cost-minimisation workflow scheduling algorithms: the Hybrid Cloud Optimised Cost (HCOC) [35] and the Deadline-MDP (Markov Decision Pro-

cess) [144]. According to the authors, the use of imprecise bandwidth estimates can strongly degrade the performance of produced schedules. They also observed that the impact is more significant on links that have low bandwidth. Deadline violation, long makespan, and high costs were the most common negative consequences that occurred in the experiments carried out by the authors. In contrast to the procedure proposed in this thesis, the authors in [35] do not provide any solution to improve the robustness of both HCOC and Deadline-MDP schedulers under uncertainty in the available bandwidth. As in the evaluation performed in this thesis, there are also comments in [35] regarding the importance of the robustness of the scheduler in dealing with imprecise bandwidth information, especially workflow schedulers for hybrid clouds.

Imprecision in input information to the scheduler has been approached mostly via reactive techniques, such as dynamic scheduling [5], rescheduling [120], and self-adjusting [17]. Reactive techniques are best suited in situations when the resource availability fluctuates considerably, such as the available bandwidth in inter-cloud links. As reactive mechanisms are based on monitoring of resource utilisation, imprecise information may also be due to the intrusion effect of monitoring tools [14]. As a result, for instance, unnecessary task migration among clouds can occur due to this monitoring overhead [16].

In order to avoid the drawbacks of reactive approaches, Batista and Fonseca presented in [16, 19] a proactive approach to deal with uncertainties in application demands and resource availability on the scheduling of workflows in grids. They proposed a scheduler based on fuzzy optimisation that does not require the active overhead of monitoring the underlying network to extract the current bandwidth estimates in order to produce the schedules. The fuzzy-based makespan-minimisation scheduler, called IP-FULL-FUZZY, is a robust scheduler for grid networks under uncertainties of both application demands and resource availability. The imprecision in input information is modelled as fuzzy triangular numbers since these numbers can assume different uncertainty levels. Although the authors compared the proposed algorithm to static schedulers for heterogeneous systems based on grid computing, the impact of imprecise estimates of the available bandwidth were not evaluated on clouds. To solve this issue, Chaves et al. adapted the IP-FULL-FUZZY in [46] to work in cloud environments and proposed the TVM-Fuzzy, a fuzzy-based scheduler that minimizes the makespan of cloud applications (modelled as workflows) under uncertain available bandwidth values. The scheduler considers the need of instantiating virtual machines before the execution of applications and it accepts information on the available bandwidth that can be furnished by real bandwidth estimators such as *pathload*, which provides intervals representing the bandwidth availability [13, 14]. The authors state that the makespan produced by the proposed TVM-Fuzzy scheduler were robust in the face of uncertainties of available bandwidth; however, in contrast to our proposal, the TVM-Fuzzy has not been evaluated in a hybrid cloud scenario.

In fact, both schedulers (the IP-FULL-FUZZY and the TVM-Fuzzy) treat the scheduling problem as an integer linear-programming (ILP) problem since the fuzzy optimisation problem is transformed into an equivalent ILP. However, as shown in [72], ILP-based scheduler usually tends to be time-consuming to produce schedule solutions, process that could take from minutes to hours when optimal (or even sub-optimal) schedules are required. This occur because the scheduling problem is, in general, an NP-Complete

problem [112], and as a consequence, the time taken by a ILP-based scheduler to find (sub)optimal solutions increases exponentially with the input size of the problem, which in this case is basically the size of the workflow (DAG) in terms of the number of nodes and edges. Actually, we can easily observe in [16, 19, 46] that both ILP- and fuzzy-based schedulers were evaluated using only DAGs of up to 26 nodes and 48 edges. According to [129], DAGs of this magnitude usually do not represent realistic applications since real applications are normally represented by large DAGs of at least 500 to 100 nodes and 1000 to 2000 edges [129]. In this case, these robust schedulers could take hours or even days to find any feasible solutions for DAGs of this magnitude [72]. Although the works [16,19,46] have introduced robust ILP-based schedulers under uncertainty of available bandwidth, there are evidences in [72] that these schedulers may not be a better choice to produce scheduler for realistic workflow applications that are usually represented by DAGs that may easily surpass, for instance, 100 nodes and 200 edges. Moreover, these robust schedulers only promote the minimisation of the makespan. Also, since resources from public cloud providers are normally leased on a pay-as-you-go basis, usually for hours of use, the literature still lacks deadline-driven robust schedulers under imprecision in input data that also minimises the monetary execution costs.

In contrast to related works, this thesis aims in making existing non-robust workflow schedulers in being able to make robust schedules that should be insensitive to the uncertainties stemming from using unreliable monitoring networking tools. Those schedules are not robust under uncertainties of available bandwidth because they compute schedules without addressing the imprecision of input data. Note that this thesis does not intended to create a new robust scheduler under those uncertainties. The procedure proposed here allows non-robust schedulers, which were not designed to address imprecision in input information, in being able to make scheduling decisions which should be robust in the face of the uncertainty of available bandwidth. Unlike the ILP-based schedulers, heuristics-based ones are not designed to guarantee the bounds of the produced solutions, and as a result, its execution time typically tends to be faster [72]. As these non-robust heuristic-based schedulers are usually used to produce quickly schedules for large DAGs previously commented, the procedure proposed in this thesis seems to be an attractive tool to improve their robustness under imprecise bandwidth estimates.

3.3.2 Bandwidth Estimation given in Ranges rather than Deterministic Values

Measurement of available bandwidth in today's cloud providers has recently been a topic of great interest [14, 82, 91, 92, 108, 116, 121–123, 137]. The concept of bandwidth is very important for present day networks, and various applications can benefit from this information to be executed adequately, especially applications modelled as workflows. At the physical layer, bandwidth refers to the width of the communication spectrum available for data transmission. However, at the network and higher layers, it refers to the data rate available on a network path or link, usually expressed in bits per second [50]. The term bandwidth is often confused with different throughput related concepts. We consider the term *available bandwidth* in bit/s (bps) as the amount of available data that can be carried

through a network connection in a given time period, or the *unused* or *spare* capacity of the link during a given time period that is available to transmit data.

Monitoring network performance of a cloud provider has recently attracted great interest in the research community. By not relying on the information advertised by the cloud provider, researchers have been using monitoring/measurement networking tools to appropriately analyse the network performance with the purpose of running bandwidth-sensitive applications appropriately [2,108,121,122]. In general, tools for network measurement can be classified into two categories [9, 79]: active probing or passive measurement. The active probing requires probing traffic to understand the network characteristics. By the help of transmitted probing packets, the available bandwidth is measured by sending dummy packets from sender node to receiver node. Active monitoring tools, however, have the side effects of flooding the network with (dummy) traffic that may directly impact the performance of other applications already in execution, besides producing imprecise bandwidth estimates due to the intrusion effects of the probing traffic [14]. On the other hand, passive measurement does not inject dummy packet in the network since it focuses on the monitoring of traffic already present in the network [9]. Although it is considered potentially very efficient and accurate approach to provide measurements of the available bandwidth [124], the passive measurement is limited to network paths that have recently carried user traffic out. Active probing, on the other hand, has the advantage of being able to explore the entire network [79, 124]. Because of this, a wide range of research conducted active probing to analyse the network performance of a cloud provider [82,91,92,108,116,121–123,137], especially to investigate the available bandwidth in intra- and inter-cloud links.

Regardless the method employed by the monitoring/measurement networking tools, the estimates of available bandwidth are very likely to be given in ranges rather than deterministic values [14]. This occurs because networking links, especially the Internet ones, are shared communication infrastructure whose available bandwidth fluctuates considerably. The available bandwidth in these links may vary unpredictably and cannot be estimated precisely in advance. Due to this, it is very likely that monitoring/measurement tools will express the end-to-end available bandwidth between a pair of hosts as a range of values. Regarding the context of workflow scheduling, it is reasonable to assume that, when a deterministic workflow scheduler is employed, the information about the available bandwidth will be taken from this range since deterministic schedulers expect deterministic values rather intervals of values. When a range of bandwidth estimates is given by the tool, we can assume, for the sake of discussion simplicity, that the median value of the range will be likely furnished as input to the scheduler to produce a schedule. However, any other information from this interval can assume the available bandwidth information furnished to the scheduler. This highlights one of the main issues faced in the scheduling problem regarding the *low quality of the information* provided as input data to the scheduler to produce a schedule [44]. Therefore, the procedure proposed in this thesis attempts to reduce the impact of using low-quality information about the available bandwidth stemming from using unreliable networking tools on the production of schedulers by deterministic schedulers.

The estimated available bandwidth given in a wide range can be a result of imprecise

measurement network tools and/or their incapacity of estimating the real value of the available bandwidth. In order to calculate and compare how large is the bandwidth ranges given by different tools, we employ the *quality of stretchiness* (QoS) τ, where $\tau \geq 1$ and $\tau \in \mathbb{R}_+$. Let $[x, y]$ be a range of bandwidth values given by a tool, where $x < y$ and $x, y \in \mathbb{R}_+$. Without loss of generality, we depict $[x, y]$ as $[x, k \cdot x]$ by assuming $y = k \cdot x$, where $k \geq 1$ and $k \in \mathbb{R}_+^*$. The calculation of the range's stretch τ is defined as follows: $\tau = \frac{kx}{\bar{m}} = \frac{2k}{(k+1)}$, where $\bar{m} = \frac{x(1+k)}{2}$ is the median value of $[x, k \cdot x]$. It is attributed the minimum stretch value when $k = 1$, because the range is reduced to a deterministic value. Therefore, if $k = 1$, then $[x, 1 \cdot x] = [x, x] = x$, which results in $\tau = 1$. Otherwise, the bandwidth estimation is provided in a range rather than a deterministic value, e.g., $k > 1$ and $[x, k \cdot x]$, meaning that $\tau > 1$. The higher the k value, the larger the interval's stretch τ, and likely the lower quality of the information about the available bandwidth employed in the workflow scheduling problem. For example, for an interval $[90, 110]$, the k value is given by $\frac{110}{90}$, which results in $\tau = 1.10$. However, for $[90, 150]$, then $k = \frac{150}{90}$ and then $\tau = 1.25 > 1.10$, meaning that the interval $[90, 150]$ is wider than $[90, 110]$. Importantly, the quality of stretchiness is only a metric to compare the stretchability of different bandwidth measurements intervals given by different tools experimented in papers. The quality of stretchiness is *not* a measurement of the quality of information (QoI) [14], which describes the degree of certainty in relation to a specific value. A bandwidth estimates with a QoI of 100% (QoI = 1), for example, means an estimate with 100% certainty. A low QoS value does not necessary means a high QoI value. For example, an ideal tool may furnish a deterministic bandwidth estimate ($\tau = 1$) but 100% imprecise (QoI = 0). A mathematic relation involving the QoS and QoI values seems to be a promising research field to improve the usability of networking tools for the workflow scheduling problem, but it is left as future work.

The works presented below describe benchmark-based experiments executed on resources leased from public cloud providers, especially on the Amazon EC2[1]. Several different types of benchmarks were carried by these works, focusing mainly on the VM deployment time, the performance of CPU, memory, and disk I/O, the storage service access, and the network bandwidth. We, therefore, attempt to summarise only the information regarding the benchmark of network bandwidth and, if possible, to provide the quality of stretchiness τ according to the interval of bandwidth measurement in intra- and inter-cloud (datacenter) links reported by the authors. Intra-cloud (or intra-datacenter) links mean the links that connect VMs hosted in the same datacenter, while inter-cloud (or inter-datacenter) links mean the links (probably Internet links) that connect VMs localised in different datacenters, which may or may not be owned by the same provider.

Wang and Ng have investigated in [137] the impact of virtualisation on the networking performance of the Amazon Elastic Cloud Computing (EC2) datacenter. They present a quantitative study of the end-to-end networking performance among Amazon EC2 instances from users' perspective. They took advantage of *TCPTest* and *UDPTest* tools, which were developed by them, to characterise the intra-cloud network performance. The authors carried out experiments over space (large number of VMs, short time interval)

[1] `https://aws.amazon.com/ec2/`

and time (reduced number of VMs, long time interval). The results reported in [137] show an available bandwidth in intra-datacenter links in the range of $[200, 900]$ Mbps (for the spatial experiment) and $[400, 900]$ Mbps (for the temporal experiment). They noticed a remarkable end-to-end delay variation and a notable bandwidth throughput instability, which negatively impacts the performance of application executions, especially those scientific applications modelled as workflows. The authors argue that the unstable network performance can dramatically skew the results of the monitoring/measurement networking tools. According to the bandwidth measurement reported in [137], we have observed that the range of bandwidth values stemming from using the *TCPTest* and *UDPTest* tools has a quality of stretchiness τ of up to 1.64.

Sanghrajka et al. have carried out a network performance benchmark using the Amazon EC2 cloud provider in [121] and the Rackspace.com cloud provider in [122]. The experiment was carried out using the networking tool *iperf*[2]. For each commercial provider, they have analysed the measurements of bandwidth and latency within and across datacenters in different geographical regions. In each cloud providers, the authors have diagnosed the presence of a high instability of bandwidth measurements in both intra- and inter-datacenter links. The results reported using the Amazon EC2 show an available bandwidth in intra-datacenter links in the range of $[500, 600]$ Mbps in US, $[500, 700]$ Mbps in Europe, and $[700, 750]$ Mbps in Asia. Regarding the bandwidth measurement in inter-datacenter links (US, Europe, and Asia), the experiment revealed an available bandwidth in the range of up to $[5, 100]$ Mbps. For the experiments using the Rackspace.com [122], the available bandwidth in links connecting VMs in the different US-located datacenters was in the range of up to $[5, 25]$ Mbps. According to the intervals of bandwidth measurement stemming from monitoring/measurement networking tool reported in [121] and [122], we have observed a quality of stretchiness τ of up to 1.20 (intra-datacenter) and 1.90 (inter-datacenter) for the Amazon EC2, and of up to 1.67 for the Rackspace.com.

Persico et al. also presented in [108] a detailed analysis of the performance of the internal network of the Amazon EC2. In contrast to the previous work, they employed a different monitoring networking tool called *nuttcp*[3]. According to the authors, the network throughput among Amazon EC2 VMs localised within the same datacenter typically reaches a higher value during a first transient period, and then settles to a lower yet stable value. For example, the authors reported a measured throughput in intra-datacenter links in the range of up to $[280, 450]$ Mbps during the first 30 seconds of the experiments; after that, the available throughput remained quite stable in the range of $[280, 320]$ Mbps. They stated that the network throughput reached a stable value only after an initial transient period of approximately 30 seconds, likely due to the network resource allocation strategy employed by the cloud provider. Therefore, they argued that applications using short-lived communications may obtain higher, although unstable network throughput. As they stated, the presence of this initial throughput spike cannot be ignored by researchers/users willing to provide an accurate view of the network performance. According to the results reported in [108], we have noted that the range of bandwidth values stemming from using the *nuttcp* tool has a quality of stretchiness τ of up to 1.24 for the first 30 seconds of the

[2] `https://github.com/esnet/iperf`

[3] `http://nuttcp.net`

experiments, dropping to approximately 1.07 after the initial transient period.

Li et al. also have carried out in [92] benchmarks in different clouds in terms of cost, VM deployment time, computation, storage, and networking. Regarding the network benchmark, they have focused on the measurement of network throughput and latency on the intra- and inter-datacenter links among VMs using *iperf*. The authors employed labels instead of real names to identify the commercial names of different providers, but according to the description of the geographical regions of the datacenter, it is likely to be the commercial Amazon EC2 cloud provider. The results reported in [92] show a measured available bandwidth in the range of $[600, 900]$ Mbps in the intra-datacenter links. Regarding the links connecting VMs localised in different datacenters, the available bandwidth was in the range of $[40, 500]$ Mbps. According to these intervals, we calculate the following τ values: 1.20 for links intra-datacenter links and 1.85 for links inter-datacenter links. It is notable that the τ value for inter-datacenter links tends to be greater than the τ value for intra-datacenter links since inter-datacenter links are normally connected through Internet links whose available bandwidth may fluctuates more than the available bandwidth of network links inside of a datacenter.

Schad et al. also have carried out in [123] network benchmarks on the Amazon EC2 provider in several datacenters in US and Europe using the *iperf*[2] to measure the network throughput among VMs. According to the results of networking performance reported by them, the available bandwidth in inter-datacenter links ranged from $[200, 800]$ Mbps in US and $[300, 900]$ Mbps in Europe. The calculated τ values of these bandwidth intervals are quite similar, 1.60 for the measurements in US and 1.50 for the measurements in Europe.

LaCurts et al. proposed in [91] an approach to deploy applications on VMs in accordance with the network performance, such as end-to-end latency and throughput. The authors proposed a network-aware placement system, called Choreo, to enforces the application placement only after the measurement of the network performance among virtual machines has been carried out. The authors employed the *netperf*[4] to measure the network and detected a large network throughput variability among VMs leased from the Amazon EC2 US-located datacenter. They reported an estimated available bandwidth in the range of $[300, 1100]$ Mbps, which resulted in the quality of stretchiness τ of approximately 1.58. The authors finalised the paper stating that today's most applications are bandwidth-intensive, and an unstable network can be a drawback for their performance, even deployed entirely in a well-provisioned commercial cloud such as Amazon EC2. Note that applications composed by dependent-tasks and modelled as workflows were not addressed by the authors as this thesis focuses immensely.

Raiciu et al. have used in [116] different tools (*traceroute*, *ping*, and *iperf*) to obtain a blueprint of the EC2 network performance and took advantage of it to properly deploy applications on clouds and optimise their performance. They reported evidences of paths between VMs of different lengths and with available bandwidth in the range of $[1000, 4000]$ Mbps, depending on VM position on the datacenter. Thought the experiment carried out, the author attempted to predict the topology of the Amazon EC datacenter is US and concluded the popular fat-tree topology [4]. To reach this conclusion, they ran

[4] `https://github.com/HewlettPackard/netperf`

ping and *traceroute* between all leased VMs and identified several situations:

- The VMs are on the same physical machine. In this case, there is only one hop between the two VMs, which is the hypervisor. Using the *iperf* networking tool, they found an available bandwidth between VMs around 4Gbps (4000 Mbps);
- The VMs are in the same rack, but not in the same physical machine. They assumed the existence of a top of the rack switch (ToR) among the two VMs. In this case, the available bandwidth was around 1Gbps (1000 Mbps);
- The VMs are in the same subnet. They assumed the existence of at least one edge switch among the two VMs. In this case, the available bandwidth was around 1000 Mbps as well; and
- The VMs are in different subnets. They assumed the existence of at least one core switch and two edge switches among the two VMs. The available bandwidth was also around 1000 Mbps in this case.

We have noted that the bandwidth measurements were higher than those reported in previous works. However, as the Amazon EC2 does not make it clear at VM renting time whether two or more VMs will be hosted on the same physical server, we can expect an imprecise bandwidth estimates in the intra-datacenter links. The reported estimated available bandwidth in intra-datacenter links was in the range of $[1000, 4000]$ Mbps, which results in the quality of stretchiness τ of approximately 1.60.

Note that most of the works above employed the *iperf*[2] tool to measure network performance among virtual machines of a cloud provider. According to [123], this tool is a modern alternative for measuring the TCP/UDP bandwidth performance. Unlike other network tools, such as the *netperf*, the *iperf* consumes less system resources, which results in more precise results [123]. Nevertheless, Batista et. al. have analysed in [14] the performance of two others monitoring/measurement networking tools, the *pathload*[5] [82] and *abget*[6] [6] tools. The authors aimed to test these tools for their adoption in the scheduling of workflow applications in grid environments. They evaluated both tools using a local controlled testbed composed of 4 hosts and 2 routers, and reported inaccuracies in bandwidth estimations given by both tools. The bandwidth estimations derived by the *pathload* were more accurate than those from the *abget* and they declared the *pathload* as the preferable tool for estimating available bandwidth to produce workflow schedules in grid environments. The authors also stated that using the *abget* to estimate the available bandwidth would provide schedules that leads to underutilisation of network links since this monitoring tool generally underestimates the available bandwidth. They concluded the paper stating the importance of considering the potential impact (imprecision) on the estimations may have due to the presence of other flows during the measurements. In contrast to this thesis, the authors do not provide any procedure in how to deal with these imprecise bandwidth estimates in the workflow scheduling problem.

[5] `https://www.cc.gatech.edu/~dovrolis/bw-est/pathload.html`
[6] `https://www.cc.gatech.edu/~dovrolis/bw-est/abget_tutorial.html`

Incidentally, works employing *pathload* and *abget* to conduct network benchmarks using resources of public cloud providers were not found in the literature. Maybe because there is no support for them in the last 10 years. The latest version of both is from 2006 versus 2016 for *iperf* and *netperf*. Nevertheless, others studies that compare tools for estimating available bandwidth including *pathload*, *abget*, *ping*, *ipferf*, and *netperf* can be seen[7] in [9, 38, 76, 125, 126].

[7]Some tools for bandwidth estimation available in `https://www.icir.org/models/tools.html`

Table 3.1: Summary table of works that provide estimates of end-to-end available bandwidth in the intra- and inter-cloud links. Note that the estimates are given in ranges rather than deterministic values.

Work	Cloud Provider	Intra- or inter-datacenter	Networking Tool	Measurement Method	Available bandwidth in range of Mbps	Median	τ
Wang and Ng [137]	Amazon EC2	Intra-datacenter	*TCP/UDP test*	Active	$[200, 900]$ spacial experiment	350 Mbps	1.64
					$[400, 900]$ temporal experiment	650 Mbps	1.39
Sanghrajka et al. [121]	Amazon EC2	Intra-datacenter	*iperf*	Active	$[600, 900]$ in US	750 Mbps	1.20
					$[500, 700]$ in EU	600 Mbps	1.17
					$[700, 750]$ in Asia	725 Mbps	1.04
		Inter-datacenter			$[5, 100]$ among US, EU, Asia	52.5 Mbps	1.90
Sanghrajka et al. [122]	Rackspace.com	Intra-datacenter	*iperf*	Active	$[5, 25]$ in US	15 Mbps	1.67
Persico et al. [108]	Amazon EC2	Intra-datacenter	*nuttcp*	Active	$[280, 450]$ initial transient period	365 Mbps	1.24
					$[280, 320]$ after transient period	300 Mbps	1.07
Li et al. [92]	NA	Intra-datacenter	*iperf*	Active	$[600, 900]$	750 Mbps	1.20
		Inter-datacenter			$[40, 500]$	270 Mbps	1.85
Schad et al. [123]	Amazon EC2	Inter-datacenter	*iperf*	Active	$[200, 800]$ in US	500 Mbps	1.60
					$[300, 900]$ in EU	600 Mbps	1.50
LaCurts et al. [91]	Amazon EC2	Intra-datacenter	*netperf*	Active	$[300, 1100]$ in US	700 Mbps	1.58
Raiciu et al. [116]	Amazon EC2	Intra-datacenter	*iperf*	Active	$[1000, 4000]$ in US	2500 Mbps	1.60

Table 3.1 summarises all the works previously described in this section. These works have reported benchmark-based experiments executed on resources leased from public cloud providers. Thus, the column *cloud provider* characterises in which provider the experiments were carried out. As we can see, most of the works have employed the worldwide mostly used the commercial Amazon EC2 cloud provider. The column *intra- or inter-datacenter* refers to whether the measurement concerning the available bandwidth between VMs were carried out between datacenters or within the same datacenter. Note that, in the experiments involving inter-cloud links, the datacenters were usually owned by the same provider, and the links between them probably included Internet links whose available bandwidth fluctuates considerably because of the shared communication infrastructure. The measurement/monitoring networking tool employed to carried out the benchmark-based experiments in order to measure the network performance is referenced by the column titled *networking tool*. As we can observe, the most commonly used tool was the *iperf*, a widely-used tool for network performance measurement purposes. Basically, *iperf* has the client and server functionality and, with this, it can create data streams to measure the throughput between the two ends in one or both directions.

The column *measurement method* highlights the approach employed by the tool to carried out the measurement of the network. As we can observe, all of them employ the acting probing approach, which sends dummy packets from sender host to receiver host to measure, for instance, the available bandwidth among them. The active probing requires probing traffic to understand the network characteristics and, therefore, it has the advantage of being able to widely explore the network, which it is not possible through passive measurement. The column *available bandwidth in range of Mbps* emphasises that the measurement of available bandwidth is given in ranges rather than deterministic values. The last column of the table τ indicates the quality of stretchiness (QoS) of the range. It illustrates that the available bandwidth in intra- and inter-cloud links may vary and it likely to be furnished in ranges and not estimated precisely as a deterministic value. The quality of stretchiness is only a metric to compare the stretchiness of different bandwidth measurements intervals given by different tools. The larger the wideness of the interval, the larger the interval's stretch τ, and likely the higher the imprecise information about the available bandwidth used in the workflow scheduling problem.

A significant number of workflow applications that run on cloud resources are network-intensive, transferring large amounts of data between these resources. The performance of such applications, especially those grouped into ensembles [73, 74], depends not just on processing and disk performance but also on the network performance between the resources on which they are deployed. Deterministic workflow schedulers assume that the estimated available bandwidth among resources furnished to them at the scheduling time is 100% precise. However, Table 3.1 shows that this assumption is not true. Tools for estimating available bandwidth produce estimates with large variability. The estimates of end-to-end available bandwidth in the intra- and inter-cloud links are given at very wide rather than very narrow intervals or even less likely in deterministic values. The larger the interval's the interval's stretch τ, the higher the imprecise information about the available bandwidth furnished to the scheduler. As shown in [24, 67, 68], schedules produced by deterministic schedulers using imprecise information about the available

bandwidth may make decisions that lead to ineffective executions of applications.

When the measured bandwidth estimates used at scheduling time do not correspond to the actual ones experienced at the execution time, the application execution may be underperforming, which means that the schedule that guided this execution is poor. In Section 3.5 we propose a procedure that attempts to reduce the impact of using uncertain information about the available bandwidth stemming from using (unreliable) networking tools on the production of schedules by existing deterministic schedulers. However, before describe it, Section 3.4 present an overview of three deterministic workflow schedulers. These schedulers were used in Section 3.6 to evaluate the effectiveness of the proposed procedure. Evaluation results, showed in Section 3.7, have indicated that, by using the procedure, the performance of these schedulers fed by imprecise bandwidth estimates underwent less degradation than that of the performance of the same schedulers without using the procedure.

3.4 An Overview of Schedulers for Hybrid Clouds

This section describes an overview of three deterministic workflow schedulers designed to be used in hybrid cloud scenarios. However, they were not designed to address inaccurate information to produce a schedule, especially imprecise bandwidth estimates stemming from using (unreliable) networking tools. A cost-aware scheduler based on particle swarm optimisation (PSO) proposed by Pandey et al. in [104] is described in Section 3.4.1. Since one of the purpose of composing a hybrid cloud environment is to address the lack of computing resources of a private cloud to satisfy necessary deadlines of application executions, Section 3.4.2 modifies the PSO-based scheduler to make it into a cost- and deadline-aware scheduler. Section 3.4.3 describes the Hybrid Cloud Optimised Cost (HCOC), a cost- and deadline-aware workflow scheduler specifically for hybrid clouds.

3.4.1 A PSO-based Cost-aware Scheduler

One class of schedulers that can be employed in hybrid clouds focuses only on cost minimisation since resources are leased on a pay per use basis. Pandey et al. proposed in [104] a workflow scheduler that employs particle swarm optimisation (PSO) [90], a population-based heuristic search algorithm. A heuristic is employed by the authors to iteratively utilise the results from PSO to produce the schedules. PSO is a popular tool due to its simplicity and its effectiveness in solving a wide range of NP-Complete problems, such as the workflow scheduling problem, at a low computational cost. However, PSO makes few or no assumptions about the problem being optimised, and therefore this technique does not guarantee the bounds for the optimality of the achieved solution.

An Overview of the Particle Swarm Optimisation

Inspired by social behaviour of bird flocking or schooling fish, PSO is composed of a population (called a swarm) of candidate solutions (called particles), and solves the problem

by moving these particles around in the search space according to a mathematical formula (fitness function). In PSO, the particles "fly" through the problem search space by "following" the best particles that currently represent the best solution. Each particle's movement is influenced by its locally best known position (named as *pbest*), but is also guided toward the best globally known positions (named as *gbest*) of the entire population. PSO does not apply evolution operators (mutation or crossover) between individuals of the population as it occurs in genetic algorithms [90]. Instead, in each iteration, all the particles individually update themselves to improve the solutions that they represent.

Algorithm 1 PSO procedure overview [90]

```
Require: An integer n
 1: procedure PSO
 2:     Create a swarm with the fixed number of particles n
 3:     for each particle of the swarm do
 4:         Initialise the particle with random solutions to the problem
 5:         Set the particles's pbest as null
 6:     end for
 7:     Set the swarm's gbest as null
 8:     while the maximum number of iterations is not reached do
 9:         for each particle of the swarm do
10:             Calculate its fitness value by using a fitness function        ▷ An abstract function
11:             if the current fitness value is better than particle's pbest then
12:                 Set the current fitness value as the new pbest
13:             end if
14:         end for
15:         Choose the best pbest between all particles of the swarm
16:         if the current pbest is better than previous gbest then
17:             Set the current pbest as the new gbest
18:         end if
19:         for each particle of the swarm do
20:             Update particle's velocity according equation
21:             Update particle's position according equation
22:         end for
23:         Increment iteration
24:     end while
        return position gbest as the solution of the problem
25: end procedure
```

The particles are assessed by a fitness function, which calculates a fitness value to evaluate the quality of the solutions given by them. Iteratively, PSO is guided to select the particle that contains the minimum (or maximum) fitness value of the entire population, and set this particle as the best candidate solution to the problem. As the position of the current best particle represents a region that potentially contains good candidate solutions, PSO greedily attempts to move the swarm toward this region. This process is repeated until the stop condition is reached, when the PSO returns the best solution achieved so far. Algorithm 1 describes an overview of the PSO procedure.

An Overview of the PSO-based Scheduler

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be a DAG and $\mathcal{R} = \{R_1, R_2, \ldots\}$ be a set of resources of a hybrid cloud. Let ρ be a particle that is represented by an array of size k, where $k = |\mathcal{V}|$ is the total

number of nodes (tasks) contained in $\mathcal{G}$. The order of the tasks in the array follows the topological order given by the depth-first search algorithm. The value assigned to each particle's dimension (array's position) is an integer number that represents the index of a resource in $\mathcal{R}$. Hence, each particle of the swarm represents a potential schedule that allocates $\mathcal{G}$ to $\mathcal{R}$. A k-dimensional particle is illustrated in Figure 3.6.

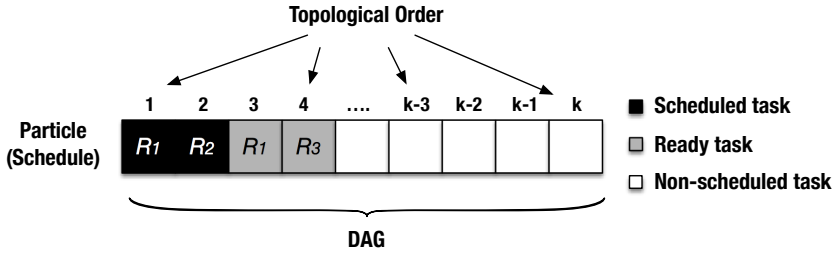


Figure 3.6: A sample of a candidate particle that represents a potential schedule for a DAG $\mathcal{G}$ to $\mathcal{R}$, where $\mathcal{R} = \{R_1, R_2, R_3\}$ is a set of resources a hybrid cloud assuming, for instance, R_2 and R_3 as resources leased on a pay-per-use basis.

Let $f : \mathbb{N}^k \rightarrow \mathbb{R}$ be the fitness function that must be minimised. This function takes a k-dimensional particle as argument, and produces a real number as output, which indicates the particle's fitness value. The aim of PSO is to find a particle ρ such that $f(\rho) \leq f(q)$ for all particles q in the swarm. It means that ρ is the particle that represents the global minimum in the swarm. The maximisation can be performed by considering the function $h = -f$ instead. Associating the fitness value as the monetary execution cost of the schedule given by ρ, PSO can achieve particles that represent low-cost schedules for the workflow scheduling problem. Algorithm 2 describes the fitness function employed by the PSO-based scheduler in [104].

Algorithm 2 Fitness Function of the PSO-based cost-aware scheduler [104]

Require: a k-dimensional particle ρ as a vector of integer numbers
1: **procedure** FITNESS FUNCTION (a particle $\rho[\,]$)
2: Calculates the overall monetary execution cost of the schedule given by the particle ρ
3: Calculates the overall monetary transfer cost of the schedule given by the particle ρ
4: **return** the overall monetary cost as the fitness value of ρ to Algorithm 1
5: **end procedure**

Algorithm 3 introduces the PSO-based scheduler. Initially, the scheduler computes some parameters, such as the average computational demands (line 1) and communication demands (line 2), before starting calling the PSO. Next, the PSO is called (on line 5) to compute the mapping of all tasks of the workflow. To validate the dependencies between the tasks, the algorithm assigns the "ready tasks" to resources according to the mapping given by PSO. A task t is labeled "ready" when its parents have completed their execution and have provided all files necessary for its execution. Depending on the number of tasks that have been completed in the current iteration, the scheduled task list $\mathcal{S}$ is updated on line 9. Once a ready task has been assigned on a resource in the current iteration, its schedule will not be changed in the next heuristic iterations (Figure 3.6). These steps are repeated until all the tasks of the workflow have been scheduled. The authors state

Algorithm 3 PSO-based Scheduling Heuristic overview [104]

```
1: Compute the average computing cost of all jobs in all resources
2: Compute the average cost of communication between resources
3: U = V                          ▷ The set of all unscheduled task, where initially is equal to V
4: S = ∅                          ▷ The set of all tasks that have been scheduled
5: Call PSO(U)
6: repeat
7:     for each "ready" task t ∈ U do
8:         Assign t to the resource r according to the solution provided by PSO's gbest particle
9:         Add the task t in the S set
10:        Remove t of the U set
11:    end for
12:    Call PSO(U)
13: until there are unscheduled tasks in U
```

that all tasks are not mapped onto the same resource, which promotes parallelism and consequently reduces the makespan of the workflow. They also state that the scheduler avoids to produce schedules that can result in the sequential execution of the workflow on the least expensive resource available.

3.4.2 A PSO-based Deadline- and Cost-Aware Scheduler

To make the cost-aware PSO-based scheduler [104] into a deadline- and cost-aware scheduler, we modified the fitness function to calculate the monetary costs of particles that only represent schedules that satisfy a given deadline. If a particle does not satisfy the deadline, the solution is not "qualified" and the particle's fitness function is given a value of infinity. This adaptation drives the scheduler in minimising the monetary cost of particles that only represent qualified solutions (makespan within the deadline). We call this scheduler as *adapted-PSO-based scheduler* and its fitness function is illustrated in Algorithm 4.

Algorithm 4 Fitness Function of the new PSO-based cost- and deadline-aware scheduler

```
Require: a k-dimensional particle ρ as a vector of integer numbers, a deadline d
1: procedure FITNESS FUNCTION (a particle ρ[ ])
2:     Calculates the makespan of the schedule given by the particle ρ
3:     if the maskepan is less or equal to the deadline d then
4:         Calculates the overall monetary execution cost of the schedule given by the particle ρ
5:         Calculates the overall monetary transfer cost of the schedule given by the particle ρ
6:     else
7:         return +∞ as the overall monetary cost of the schedule given by ρ to Algorithm 1
8:     end if
9: return the overall monetary cost as the fitness value of ρ to Algorithm 1
10: end procedure
```

3.4.3 A Heuristic-based Deadline- and Cost-Aware Scheduler

The Hybrid Cloud Optimized Cost (HCOC), proposed by Bittencourt and Madeira [35], is a workflow scheduler for hybrid clouds that attempts to minimise the monetary cost of the execution of the workflow application while meeting specified deadlines for this execution. The HCOC decides which resources should be leased from the public cloud and aggregated

to the private cloud to provide sufficient processing power to execute a workflow within a given execution time. The HCOC starts scheduling the workflow in the private cloud first because those resources are assumed to be free of charge. If the predicted makespan exceeds the specified deadline, then tasks are selected to be rescheduled in the public cloud. While executing every task of the workflow locally may delay the execution, on the other hand, executing all tasks in the public cloud may result in prohibitive costs. Therefore, the algorithm tries to balance the use of private resources with the ones available from the public cloud in a pay-per-use basis. The HCOC scheduler works into two main steps:

1. Schedule the workflow in a private cloud; and
2. If the makespan of the workflow is larger than the specified deadline, then
   a) Select tasks to reschedule;
   b) Select resources from the public clouds to compose the hybrid cloud $\mathcal{H}$; and
   c) Reschedule the selected tasks in $\mathcal{H}$;

This initial schedule considers only the private resources $r \in \mathcal{R}$ to check if they already satisfy the deadline. If the deadline is not satisfied, the algorithm starts the process of deciding which resources it will have to request to the public cloud. This decision is based on performance, cost, and the number of tasks to be scheduled in the public cloud. To make the initial schedule, the HCOC employs the PCH scheduling algorithm.

1. Initial Step in a Private Cloud The HCOC accepts different scheduling algorithms to conduct the initial schedule in a private cloud, but the HCOC was proposed[8] using the Path Clustering Heuristic (PCH) [29]. PCH is a scheduler that was designed to minimise the makespan of the workflow. The baseline of the PCH is to create groups of nodes (*clusters of non-parallel tasks*) in a way that a task cannot be in two clusters at the same time. Each cluster is a path in the DAG. Tasks in the same cluster are scheduled in the same resource and, as a consequence, the communication overhead between theses tasks, are null. By creating clusters of non-parallel tasks, the PCH scheduler avoids communication overheads, and therefore minimises the makespan of the workflow.

Before starting the scheduling, the PCH algorithm calculates some attributes calculated for each task: *Priority* ($\mathcal{P}$), *Weight* (Computation Cost), *Communication Cost*, *Earliest Start Time* (EST) and *Estimated Finish Time* (EFT). All the information necessary to compute these attributes is given by the programming model or by the infrastructure. To compute these attributes, the PCH algorithm assumes a fake homogeneous private cloud system composed of an unbounded (infinite) number of resources. These resources have the power of the best processor available in the real private cloud system, and all links have the highest bandwidth available in the real private cloud system. Each task is assumed to be scheduled on a different resource on this fake private cloud system to make the computation of these attributes values for each task. After this initial fake scheduling, the PCH works into two main steps[9]:

[8]The HCOC adopts the PCH to produce the initial schedule; however, other scheduler could be used instead such as the Heterogeneous Earliest Finish Time (HEFT) scheduler [130].

[9]For more details concerning the PCH scheduler, please, see [29].

Task selection and clustering: In this step, the PCH creates the clusters of nodes. The first node (task) n_i selected to compose a cluster c_k is the current unscheduled node with the highest priority $\mathcal{P}$. Then, PCH starts a depth-first search on the DAG starting on n_i, always selecting the not scheduled successor $n_s \in suc(n_i)$ that has the highest $\mathcal{P}_s + EST_s$ and adding it to the cluster c_k, until n_s has no unscheduled successors. Each cluster represents a path in the DAG. The first cluster is the DAG's critical path.

Resource selection: The resource selection step is carried out after each clustering step. A cluster c_k is scheduled on the resource that gives the smallest EST for the successor of c_k. The first cluster has no predecessors and no successors, since it starts on the DAG's first node (entry) and ends on the last one (exit). Hence, having no successors, it is scheduled on the resource hat gives the smallest EFT for its last node. The other clusters have only one successor and this successor is already scheduled, since, by construction, the last node of the cluster does not have unscheduled successors. After the scheduling of each cluster, the weights, ESTs and EFTs are recomputed. If the current resource already has a cluster of the current DAG, the tasks are sorted in descending order of priority to obey the precedence constraints, avoiding deadlocks.

2. Rescheduling Step in a Hybrid Cloud After the initial schedule has been generated by the PCH, the HCOC checks if resources from the public cloud will be needed to be leased to join the private resources to compose the hybrid cloud. If the makespan given by the PCH is larger than the given deadline, the HCOC selects the task (node) n_i that is currently scheduled in the private cloud and has the largest $\mathcal{P}_i$ as the potential task to be rescheduled in the public cloud. The algorithm repeats these steps until the deadline is met or a certain number of iterations is reached.

Algorithm 5 HCOC Algorithm Overview

procedure
  $\mathcal{R}$ is the set of all resources in the private cloud
  Schedule $\mathcal{G}$ in the private cloud using PCH
  while $makespan(\mathcal{G}) > \mathcal{D}$ AND $iteration < size(\mathcal{G})$ **do**
    $iteration = iteration + 1$
    select node n_i such that its priority $\mathcal{P}_i$ is maximum AND $n_i \ni \mathcal{T}$
    $\mathcal{H} = \mathcal{R}$
    $\mathcal{T} = \mathcal{T} \cup t$
    $num\_clusters$ = the number of clusters of nodes in $\mathcal{T}$
    while $num\_clusters > 0$ **do**
      Select resource r_i from the public clouds such that $\frac{price_i}{num\_cores_i \times p_i}$ is minimum AND $num\_cores_i <= num\_clusters$
      $\mathcal{H} = \mathcal{H} \cup \{r_i\}$
      $num\_clusters = num\_clusters - num\_cores_i$
    end while
    for each $n_i \in \mathcal{T}$ **do**
      Schedule n_i in $h_j \in \mathcal{H}$ such that EFT_i is minimum
      Recalculate the jobs attributes for each job
    end for
  end while
end procedure

Algorithm 5 illustrates the HCOC scheduling algorithm. The second line of Algorithm 5 makes the initial schedule using the PCH and considering only resources in the private cloud. After that, while the deadline continues being violated, the algorithm iterates until the deadline is met or the number of iterations is equal to the number of nodes in the DAG (lines 4 to 19). Inside this iteration, the algorithm selects the node n_i such that $\mathcal{P}_i$ is maximum and n_i is not in the current rescheduling group $\mathcal{T}$ (line 6). Then, in line 8, node n_i is added to the set $\mathcal{T}$, which is composed of all nodes being rescheduled from the private cloud to the public cloud. In line 9, the algorithm computes the number of clusters of nodes in $\mathcal{T}$, which will determine how many resources (or cores) will be requested to the public cloud. After that, an iteration is repeated until the number of selected cores reaches the number of clusters in $\mathcal{T}$, always selecting the public cloud resource r_i which gives the smallest $\frac{price_i}{num\_cores_i \times p_i}$ with $num\_cores_i <= num\_clusters$. The selected resource is added to the public cloud resources pool $\mathcal{H}$ in line 12. With the resources selected, the algorithm schedules each node $n_i \in \mathcal{T}$ in the resource $h_j \in \mathcal{H}$ that results in the smallest EFT_i.

The use of incorrect estimates of the available bandwidth may steer deterministic schedulers, such as the HCOC and adapted-PSO-based, in producing poor schedules, which may violate the deadline stablished for the execution, besides surpass the budget. To minimise the production of poor schedules under uncertainty of bandwidth availability, we present in the next section the proposed procedure that attempts to reduce the impact of using uncertain information about the available bandwidth on the production of schedules by deterministic schedulers.

3.5 Handling the Available Bandwidth given in Ranges when using Deterministic Schedulers

The previous section described three different types of deterministic workflow schedulers that do not address the inaccuracies of input data to produce schedules, which imply them in producing non-robust schedules under such inaccuracies. The use of non-robust schedules can cause applications to malfunction if input data is subject to have variability while the application is running. This section focuses on ameliorating the production of schedules under imprecise bandwidth measurements stemming from using networking tools that are incapable of accurately estimating the available bandwidth at the application's scheduling time. Schedulers that are robust for low-quality of the precision of input data are of paramount importance to minimise the production of poor schedules, which usually impose the underestimate the makespan and monetary cost of the execution of applications. [19, 34].

As stated early in this chapter, the input data for a deterministic workflow scheduler can be separated in two sets: the first one with information about the application components and their dependencies, and the second one with information about the performance of the resources (processors and links) in the target system. The information in the first set can be obtained either from the workflow programming model, where the user is responsible for estimating the "size" of each task (DAG node) and each

data dependency (DAG edge); or from a data base of executions [63]. The information about resource performance can be obtained from its hardware characteristics as well as from benchmarks [82, 91, 92, 108, 116, 121–123, 137]. An important aspect of both sets is that they are prone to inaccuracy, introducing uncertainty issues to the scheduling process [14, 16, 18, 19, 24, 30, 31, 46, 68]. As we pointed out in Section 3.3.2, the bandwidth measurements are mostly given in large ranges rather than smaller ranges or even as deterministic values. The higher the interval of the available bandwidth measurements, the higher the imprecision about the bandwidth measurements furnished as deterministic values to a deterministic scheduler.

This section is organised as follows. Section 3.5.1 presents the motivation of the proposed proactive procedure that deflates the bandwidth measurements before push it to the scheduler, while Section 3.5.2 describes the that the deflating computation is carried out using information from previous executions of a target workflow carried out by a specific scheduler. The algorithm is discussed in Section 3.5.3, and the computation to deflate a bandwidth measurement is presented in Section 3.5.4.

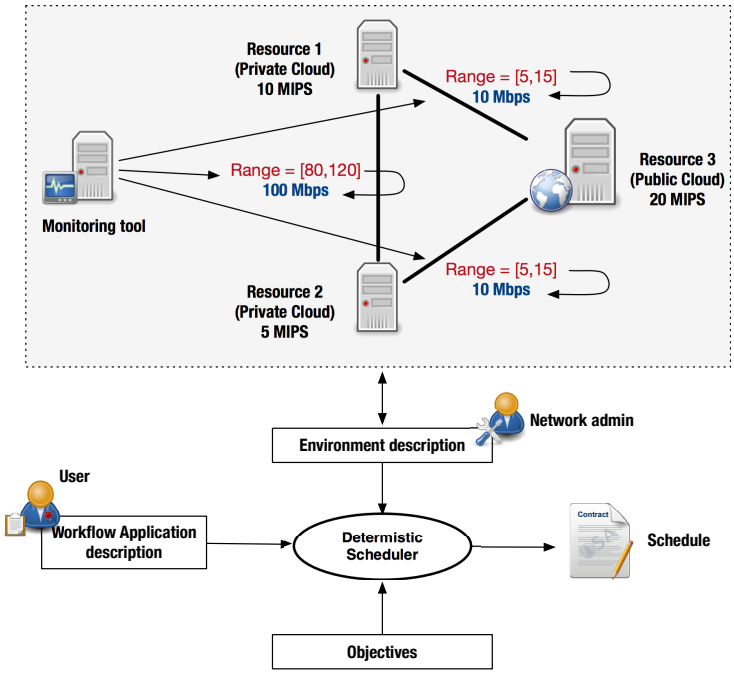


Figure 3.7: Gathering the information about the target system. It is used as input to the scheduler. The median value of each bandwidth interval is taken as a deterministic value.

3.5.1 Motivation of the Proposed Proactive Procedure

Estimations of available bandwidth are quite often given in ranges rather than as deterministic values [14, 82, 91, 92, 108, 116, 121–123, 137]. From this premise, let $[x, y]$ be a range of values representing the available bandwidth measurement given by a monitoring/measurement networking tool, where $x < y$ and $x, y \in \mathbb{R}_+$. Without loss of generality, assuming $y = kx$, we depict $[x, y]$ as $[x, kx]$, where $k \geq 1$ and $k \in \mathbb{R}_+^*$. Also, let $\bar{m} = \frac{x(1+k)}{2}$ be the median value of this interval. For the sake of discussion simplicity, when a measurement bandwidth interval $[x, kx]$ is the output of a networking tool, we assume $\bar{m}$ as

the deterministic value furnished to the scheduler representing the available bandwidth of the corresponding interval. Figure 3.7 illustrates an example of how the description of the environment execution system used as an input data to the scheduler. Although the information provided is represented as deterministic values, they are prone to a high degree of uncertainty. In this study, we considered only the uncertainties of available bandwidth.

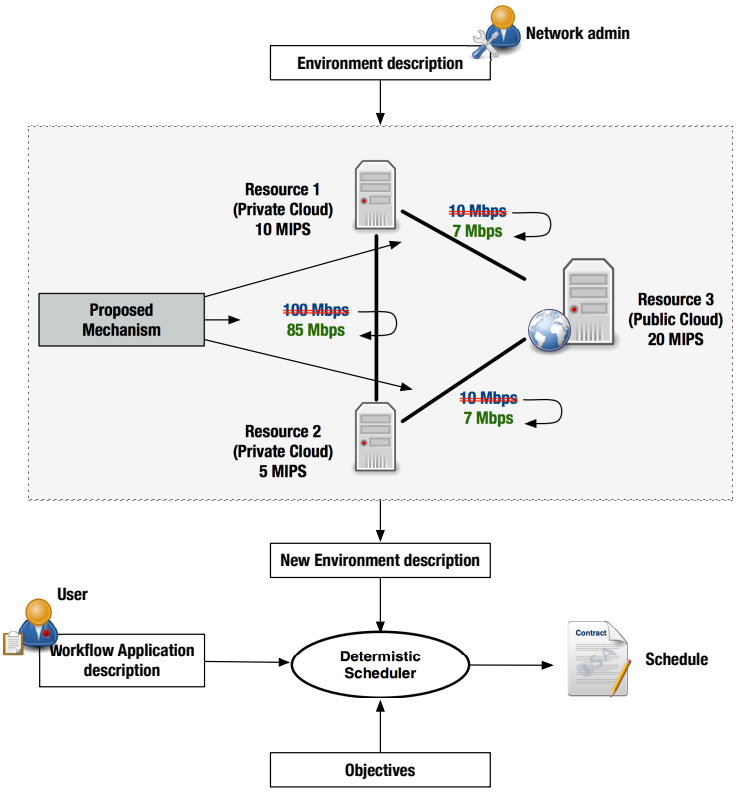


Figure 3.8: Once the information about the target system has been collected by the network administrator, the proposed mechanism (procedure) will deflate the bandwidth estimates.

Robustness to uncertainty about the information of the available bandwidth has been neglected and this can have a negative impact on the production of schedules on clouds [24]. On the one hand, overestimating the available bandwidth yields longer data transfer times than expected, which increases the makespan and can potentially exceed the established deadline. Also, this can lead to increased costs due to the need for longer rental periods, which can surpass the defined budget. On the other hand, underestimating the available bandwidth can also lead to unnecessary leasing of processing power; however, this does not call for a budget increase. Indeed, the precision of the input data that are provided to the scheduler is essential to the effectiveness of the schedule that is generated by the scheduling algorithm.

A proactive way to prevent the production of poor schedules due to the use of an imprecise bandwidth value is to apply a deflation on this value. Deflecting the available bandwidth value before used as an input for the scheduling process is a "trick" way that this thesis approaches to furnish a certain level of robustness for schedules that are produced using schedulers that were not designed to address inaccurate bandwidth information.

Proactively, this deflation is performed in the hope that, during the execution of the workflow, the real available bandwidth will not reach (due its variability) a value lower than that used to produce the schedule; otherwise, the performance of the workflow is degraded and the schedule becomes non-robust. Therefore, instead of using imprecise bandwidth measurements directly provided by monitoring networking tools, schedulers will use as input data a deflated value of the original measurements in an attempt to proactively mitigate the production of poor schedules. The deflating factor is computed by the proposed procedure through multiple linear regression using historical information of previous workflow executions, which is detailed in the next section.

Figure 3.8 illustrates the applicability of the proposed procedure after the information about the target system has been gathered by the network admin. The proposed procedure deflates the information about the available bandwidth of each link before the scheduling process starts. The proposed procedure is a proactive way to mitigate the production of misleading, poor schedules by schedulers that were not designed to deal with imprecise bandwidth estimates. The procedure of how to calculate this deflation is shown in the following section.

3.5.2 A Proactive Mechanism for Deflating Deterministic Estimates of the Available Bandwidth

The estimate of the available bandwidth furnished by monitoring tools as an input to the scheduler may deviate from that measured by a expected percent error p at the execution time of the application [14]. For example, in a scenario in which the expected uncertainty of the available bandwidth is $p = 50\%$, a data transfer that is initially estimated by the scheduler to take 30 seconds could take between 15 and 45 seconds during the execution of the workflow. To quantify such uncertainties, the mechanism employs the quality of information (QoI) model presented in [44]. The QoI index expresses the level of confidence of a available bandwidth estimation. A value of 100%, the maximum QoI value, means that there is no doubt about the estimation ($p = 0$), whereas values lower than 100% indicate that estimations are not precise ($p > 0$). The lower the QoI value, greater the uncertainty p of an estimation. The percent error p used by the mechanism is given by the expression: $p = 100\% - QoI$, where $0 \leq p \leq 1$. Thus, let b be the estimated deterministic value of the available bandwidth and p be the expected uncertainty of b. Accordingly to this model, during the execution of the workflow, the inter-cloud available bandwidth value may experience a variation from $b - p\%$ to $b + p\%$. Most schedulers in the literature neglect the variability p present in the bandwidth estimates, and as a result, the produced schedules may have their performance degraded as p increases. For this thesis, it is assumed that p is extracted from a database which is fed by the broker while the workflow applications are executed in hybrid clouds

Adaptive scheduling, dynamic scheduling and self-adjusting scheduling have all been proposed in the literature [5, 17, 120] to deal with fluctuations of resource availability. All of these schemes were designed to minimise the makespan. Resource monitoring and task migration are used in these approaches to react to fluctuations in the environment state. However, continuous monitoring can increase the actual uncertainty level due to the in-

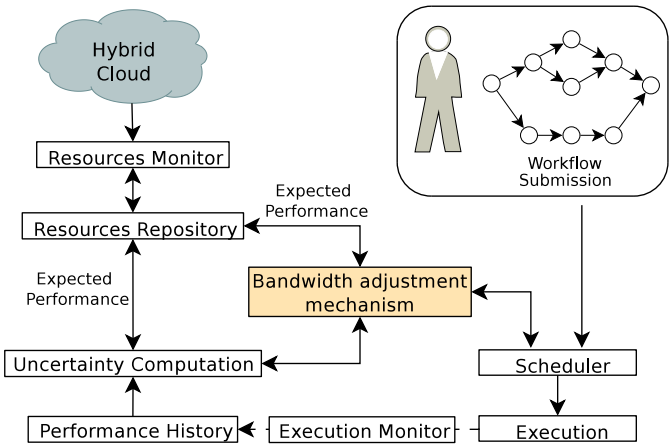


Figure 3.9: Scheduler bandwidth adjustment mechanism (procedure).

trusion effect, while unnecessary task migration can increase overhead, therefore enlarging the makespan besides costs. Traditional prediction methods, such as time series, can be used to predict the available bandwidth. However, although such predictions may be a good representation of the variability of the available bandwidth for a specific period, they would not represent the impact of the bandwidth estimates on the predicted makespan and costs of workflow executions. Thus, a procedure that correlates the uncertainty in the available bandwidth information furnished to the scheduler with the produced schedule has not been proposed.

A proactive procedure to minimise the negative impact of using imprecise estimate of the available bandwidth in the production of workflow schedules is proposed here. The proposed procedure produces a deflating multiplier value that is applied to the measured available bandwidth before being furnished to the scheduler. According to an expected uncertainty p (extracted from a database) of the estimate b (given by the networking monitoring tools), the proposed mechanism applies a reduction factor $\mathcal{U}$ in b with the goal of attempting to mitigate that a misleading bandwidth estimate b from affecting the performance of the workflow execution. By furnishing a deflated b, robust schedules that are insensitive to the uncertainty p of b *are expected* to be produced by schedulers that neglects the presence of p in b. The proposed procedure will not inflate b since this approach is left as future work. Importantly, by being a proactive approach, the mechanism does not apply any changes to the scheduler, i.e., it is just a procedure that assists the scheduler in producing robust schedules under imprecise bandwidth estimations. Note that the proposed procedure may imply the scheduler in producing worst schedules, i.e., achieving expensive schedules while the scheduler is configured to minimise the costs of the workflow execution. However, in the majority of the cases, as discussed in Section 3.7, the schedules produced using the propose procedures underwent less degradation in its performance (cost increase, for instance) than those produced without employing the proposed procedure as the variability of the available bandwidth increases.

The computation of $\mathcal{U}$ value employs information from previous executions of the target workflow (Figure 3.9). Thus, for each type of scheduler used in the scheduling process, workflow application, and a deadline value, the proposed mechanism uses previously calculated $\mathcal{U}$ values, the available bandwidth estimates and the error that occurred in

previous estimates of the makespan and cost. Regarding the uncertainty values, they can be derived from benchmarks or historical data of bandwidth availability in the network links obtained from monitor nodes [121, 122, 139].

3.5.3 Computation of the Deflating $\mathcal{U}$ Value

The proposed procedure works between the bandwidth estimation/monitoring tool and the scheduler. It takes the available bandwidth value provided by the tool, applies the deflating $\mathcal{U}$ value to it, and furnishes the deflated bandwidth value as an input to the scheduler. Historical information about the previously computed $\mathcal{U}$ values and available bandwidth estimates are stored in the database, as well as the differences between previously predicted and measured makespan and cost values of the workflow execution. A multiple linear regression (MLR) procedure is employed over a subset of the database to finally compute the $\mathcal{U}$ value. Details of the creation of the data set (also called in this document as database or historical information) are described in Section 3.6.5.

Let $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ be a data set of previous executions of a specific workflow application represented by the DAG $\mathcal{G}$. These executions were planned via the schedules given by the deterministic scheduler $\mathcal{S}$. Each row of the data set $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ is represented by 6-tuples $\langle b, \mathcal{U}, \mathcal{D}, p, m, c\rangle$, where the first three elements are information used to produce the workflow's schedule, while the others are obtained at the workflow's execution time. The elements b and $\mathcal{U}$ are, respectively, the measured available bandwidth at scheduling time and the calculated $\mathcal{U}$ used to deflate the respective b value. The third element is optional and represents the deadline value for the production of a schedule for $\mathcal{G}$. The deadline required $\mathcal{D}$ is stored in $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ when $\mathcal{S}$ is a deadline-aware scheduler; otherwise $\mathcal{D}$ assumes a value of infinity. The fourth element is the p value, which represents the maximum percentage error detected between the estimate b (given at scheduling time by the monitoring tools) and the observed available bandwidth during the execution of $\mathcal{G}$. The observed makespan (execution time) of the workflow is represented by the fifth element m, while the real cost of this execution is regarded as the sixth element c. It is stored in $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ only information concerning qualified executions, i.e., executions where the makespan of the workflow less than or equal to the deadline value.

The computation of the deflating $\mathcal{U}$-value involves the construction of a data subset $\mathcal{H}'_{\mathcal{G},\mathcal{S}} \subseteq \mathcal{H}_{\mathcal{G},\mathcal{S}}$ that is used as input to the MRL. Each row of the subset is represented by 3-tuples $\langle b, p, \mathcal{U}\rangle$. The subset is constructed in three steps. First, for all 6-tuples in $\mathcal{H}_{\mathcal{G},\mathcal{S}}$, those that contain b and p are selected. Second:

- If $\mathcal{S}$ is the PSO-based scheduler, the 6-tuple that contains the smallest cost c value (the cheapest execution), since $\mathcal{S}$ has the objective of the minimisation of costs;
- If $\mathcal{S}$ is the adapted-PSO-based scheduler or HCOC scheduler; the 6-tuple that contains a deadline value $\mathcal{D}$ and the smallest makespan m value (the fastest execution). If a tie occurs, the tuple that contains the smallest cost c is selected, since both schedulers attempt to minimise costs while satisfing the deadline;

Lastly, of the selected 6-tuple, the $\mathcal{U}$ value is taken to finally compose the 3-tuples $\langle b, p, \mathcal{U}\rangle$ for subset $\mathcal{H}'_{\mathcal{G},\mathcal{S}}$. Each 3-tuple $\langle b, p, \mathcal{U}\rangle \in \mathcal{H}'_{\mathcal{G},\mathcal{S}}$ means that, when the bandwidth

Algorithm 6 Computation of the coefficients a_1, a_2, and a_3 for the equation $f(x, y)$

```
1: H_{G,S} = Database of previous executions of DAG G
2: S = the workflow scheduler used to create the database H_{G,S}
3: H'_{G,S} = ∅
4: d = deadline value
5: for each (b', p') pair existing in H_{G,S} do
6:     ST = None                                   ▷ ST = Selected tuple
7:     for all 6-tuples ⟨b,U,D,p,m,c⟩ ∈ H_{G,S} as CT do   ▷ CT = Current tuple
8:         if CT.b == b' AND CT.p == p' then
9:             if ST == None then
10:                ST = CT
11:            end if
12:            if S == "PSO-based" AND CT.m < ST.m then
13:                ST = CT
14:            else if S == "adapted-PSO-based" OR S == "HCOC" AND CT.D == d then
15:                if CT.m < ST.m then
16:                    ST = CT
17:                else if CT.m == ST.m AND CT.c < ST.c then
18:                    ST = CT
19:                end if
20:            end if
21:        end if
22:    end for
23:    U' = Take the U value from the selected 6-tuple represented by ST
24:    H'_{G,S} = H'_{G,S} ∪ ⟨b', p', U'⟩
25: end for
26: Call a multiple linear regression by using the H'_{G,S} as an input data set and set the function f(x, y) =
    a_1 x + a_2 y + a_3 as a formula to fit the data
27: Store the coefficient values a_1, a_2, and a_3
```

estimate b had an expected uncertainty of up to p and $\mathcal{S}$ was the scheduler used to produce the schedules for $\mathcal{G}$, $\mathcal{S}$ was able to produce better results when the b was deflated by $\mathcal{U}$. Therefore, the calculation of the $\mathcal{U}$ value is carried out by a multiple linear regression (MLR) procedure, which uses the subset $\mathcal{H}'_{\mathcal{G},\mathcal{S}}$ as input data. The MLR uses this subset to compute the values of the coefficients a_1, a_2 and a_3 for the equation $f(x, y) = a_1x+a_2y+a_3$. In this equation, x is a variable that represents the predicted uncertainty in the available bandwidth, while y is an independent variable that represents the currently available bandwidth. Lastly, the deflating value is given by $\mathcal{U} = f(x, y)$. The construction of the subset and the computation of the coefficients a_1, a_2, and a_3 are illustrated in Algorithm 6.

3.5.4 The Use of the Procedure

When an application workflow $\mathcal{G}$ is about to be scheduled, the $\mathcal{U} = f(x, y)$ value is computed through the p and b obtained from the network monitor tools [14, 121, 122, 139]. Making $x = p$ the currently predicted uncertainty, and $y = b$ the currently estimated available bandwidth, a percentage reduction factor $\mathcal{U} = f(x, y)$ is computed for the next schedule computation. Therefore, for each workflow scheduler and an application workflow, the procedure requires a set of historical data for training before the multiple linear regression approach can be applied for the next scheduling procedure.

3.6 Evaluation Methodology

This section describes the methodology of the evaluation of the efficacy of the proposed procedure to make existing non-robust workflow scheduler under uncertainty of available bandwidth in links of a hybrid cloud. The performance of workflow schedulers used in this study is evaluated in the publications in which they were originally presented [35,104]. The objective of this section is to evaluate the proactivity of the proposed procedure in being able to adequately deflate the bandwidth estimates given by networking monitoring tools in accordance with an expected uncertainty of such measurements. Therefore, the purpose of the experiment is to make existing schedulers to use deflated values of bandwidth estimates, given by the proposed procedure, in order to proactively reduce the performance degradation of produced schedules caused by bandwidth variabilities. By doing this, the procedure attempts to intensify the level of robustness of schedules produced by non-robust schedulers under imprecise bandwidth estimations since these schedulers were not designed to address such inaccuracies to produce schedules.

3.6.1 Workflow Applications

The proposed procedure is evaluated using simulations with synthetic workflows from the workflow generator gallery[10]. The gallery contains synthetic workflows that were modeled using patterns taken from real applications, such as Montage, LIGO and CyberShake. Workflows with different sizes such as 50, 100, 200 and 300 tasks (DAG nodes) are employed. For each workflow application and for each workflow size, 100 different workflows were generated differing in the computational demands (weights of the DAG nodes) uniformly distributed in the interval $[1, 10]$, and the communication demands (weights of the DAG edges) uniformly distributed in the interval $[50, 60]$. The set of synthetic workflows contains 3 types of workflow applications, 4 different workflow sizes, and 100 workflows per application and size, resulting in a total of $1,200$ synthetic workflows.

Table 3.2: Hybrid cloud environment

Provider	Type	Core	Core Performance	Price per time unit	Data transfer out price
A_1 (private)	Small	1	1.0	\$0.00	\$0.00
A_2 (private)	Small	1	1.0	\$0.00	\$0.00
	Medium	1	1.5	\$0.00	\$0.00
A_3 (private)	Small	1	1.0	\$0.00	\$0.00
	Medium	1	1.5	\$0.00	\$0.00
	Large	2	2.0	\$0.00	\$0.00
B (public)	Small	1	2.75	\$0.104	\$0.01
	Medium	2	2.75	\$0.207	\$0.01
	Large	4	2.75	\$0.415	\$0.01
	X-Large	8	2.75	\$0.829	\$0.01
C (public)	Small	1	1.0	\$0.06	\$0.12
	Medium	1	2.0	\$0.12	\$0.12
	Large	2	2.0	\$0.24	\$0.12
	X-Large	4	2.0	\$0.48	\$0.12
	XX-Large	8	3.25	\$0.9	\$0.12

[10] `https://confluence.pegasus.isi.edu/display/pegasus`

3.6.2 Hybrid Cloud Scenario

Hybrid cloud scenarios are shown in Table 3.2. They employ resources and prices based on the configurations of the Amazon EC2[11] and Google Compute Engine[12] cloud providers. To simulate the availability of private resources, we use three different sizes of the private cloud to simulate situations in which private resources are not sufficient to handle the computational demands of the workflow execution, and the leasing of public resources becomes necessary to meet with the application deadline.

3.6.3 Simulator/Calculator

Figure 3.10 shows an overview of how the simulations were performed. The HCOC and PSO-based schedulers were implemented in Java according to their descriptions in [35] and [104], respectively, and the adapted-PSO according to the adjustments described in Section 3.4.2. The data inputs for the scheduler are a DAX file, a VM file and an estimated value of the available inter-cloud bandwidth b. The DAX file contains the description of the workflow in XML format, while the VMs file contains information about the hybrid cloud (Table 3.2). For the HCOC and adapted-PSO-based schedulers, a deadline value for the workflow execution is also provided as input data. After the schedule is produced by the scheduler, it is used as an input to a simulator (calculator), which simulates the schedule and calculates the metric values such as: the makespan (workflow execution time) and the monetary execution costs. Note that the simulator employed in this evaluation is not a traditional event-oriented simulator, but rather a calculator that competes the scheduling events extracted from a given schedule according to the given information about the cloud resources. For more details regarding the simulator, please see Appendix A. The words *simulator* and *calculator* will be synonymously used hereinafter. The estimated inter-cloud available bandwidth value b given by the monitoring/measuring tools is deflated by the factor $\mathcal{U}$, and the result is used by the scheduler; $\mathcal{U} = 0$ means that no reduction is applied on b, while $\mathcal{U} = 5$ means that the deflated bandwidth is 95% of the value b provided at the scheduling time.

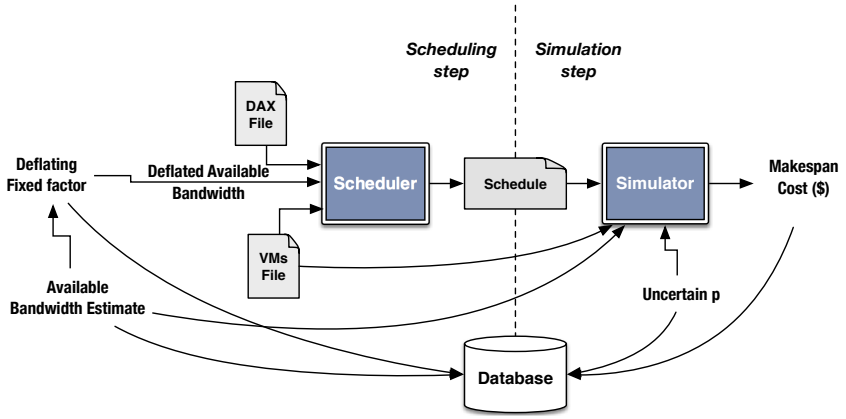
The simulator uses the uncertainty value p of the inter-cloud bandwidth b to simulate the variability in the available bandwidth. In other words, every time the simulator needs to use the bandwidth estimated b outputted from the monitoring tools, it takes a new value that is uniformly distributed in the interval $[b - p\% \,; b + p\%]$, simulating, therefore, the uncertainty (or the variability) of b.

3.6.4 Experimental Parameters

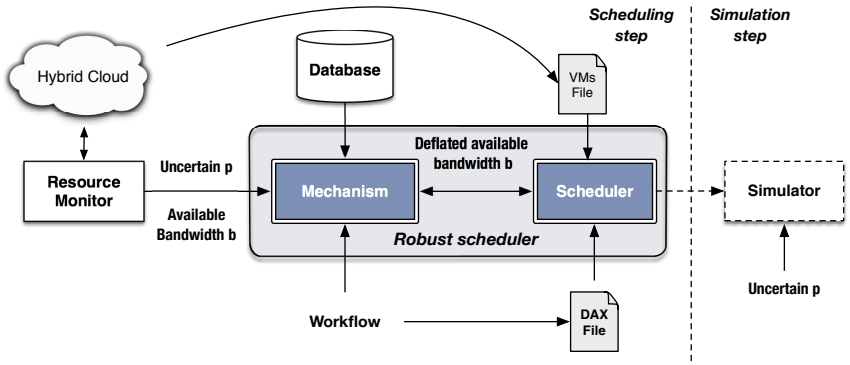
A broad space of parameter values, including deadlines, was used in cases ranging from tight deadlines, which implies that a small number of workflows can be completed with private resources, to more loose deadlines, which implying that almost all of the workflows can be completed using only private resources. In addiction, three levels of cloud utilisation were employed, ranging from an idle private cloud, in which most of the executions

[11] `http://aws.amazon.com/ec2/`

[12] `https://cloud.google.com/compute/`



(a) First step of the procedure evaluation



(b) Second and third steps of the procedure evaluation

Figure 3.10: Overview of the steps of the proposed procedure (mechanism) employed on the experimental evaluation.

could be performed without violating the deadline, to a busy private cloud, which calls for the use of a public cloud to accomplish most of the executions before the deadline.

The deadline values, $\mathcal{D}$, varied from $T_{max} \times 2/7$ to $T_{max} \times 6/7$ in $1/7$ steps, where T_{max} is the makespan of the least expensive sequential execution of all the DAG nodes on a single resource. Deadlines of $T_{max} \times 1/7$ resulted in only disqualified solutions (makespans greater than the deadline value) for all executions. Schedules for deadlines of $T_{max} \times 7/7$ can be trivially achieved by scheduling all of the tasks sequentially in the least expensive resource and were not generated in this study. The divisor 7 was chosen only to assess the evolution of the deadline, from a short deadline to a large one, having no strict relation to the proposed procedure. So, other dividers could also be used instead.

Inter-cloud bandwidths of 10 to 60 Mbps and intra-cloud bandwidths of 1 Gbps were considered. We assume that the intra-cloud bandwidth is greater than the inter-cloud bandwidth, which is a reasonable assumption in real hybrid cloud environments. The available bandwidth values in the inter-cloud links were based on measurements of TCP connections for the Amazon EC2 cloud [121] and the Rackspace.com cloud [122].

3.6.5 Experimental Setup

The procedure was evaluated in three steps. In the first step, a database that contains historical information about the workflow execution is produced for each type of workflow, deadline value, workflow size, available bandwidth, and scheduler (Figure 3.10a) using a fixed deflating bandwidth $\mathcal{U}$. In the second step, the coefficient values of the MLR equation are generated using information from the database. In the third and last step, the MLR procedure is applied to deflate the bandwidth value, which is used as an input to the scheduler (Figure 3.10b) when a new workflow is set to be scheduled.

Initially, we assessed how using the procedure with a fixed $\mathcal{U}$ factor ameliorates the negative impact of imprecise information about the inter-cloud available bandwidth on the execution of the workflow. To investigate the results of the produced schedules, fixed bandwidth deflating factors $\mathcal{U} \in \{0, 5, 15, 25, 35, 45, 50\}$ were used. The choice of the fixed $\mathcal{U}$ values was based on experiments performed for this purpose. Results indicated that when the measured available bandwidth values were deflated by more than 50%, the scheduler did not find qualified solutions within the established deadline in most of the conducted experiments. For this reason, $\mathcal{U}$ values less or equal than 50% were used in this study. For each value of $\mathcal{U}$, the percent error p ranged from 10% to 99% in order to assess the impact of $\mathcal{U}$ on the produced schedule in the presence of p during the application execution time. For example, given a $\mathcal{U}$ value, a percentage error p was introduced into the simulation step to derive the makespan and costs and store it into the database. The simulation results were stored into the database. The first step of the experimental evaluation is illustrated in Algorithm 7

In the second step, based on the type of workflow and the scheduler, the values of the coefficients of the multiple linear regression (MLR) equation are computed using Algorithm 6. The MLR derives the values of the coefficients a_1, a_2 and a_3 in the equation $f(x, y) = a_1x + a_2y + a_3$. By checking the resource monitor and making $x = p$ the currently predicted uncertainty and $y = b$ the currently available bandwidth, a deflating factor $\mathcal{U} = f(x, y)$ can be computed and used as an input to the scheduler in the last step of the experimental evaluation. In the last step, new simulations were run for each type of workflow to evaluate the behaviour of the proposed procedure. For all $\mathcal{U} = f(x, y)$ and p values, average values were computed considering only the schedules that results in makespan values lower than the deadline, i.e., qualified executions.

Importantly, the creation of data subset is strictly driven to the objective of the scheduler. In this experimental evaluation, we apply the heuristic showed in the Algorithm 6, however others heuristics can be applied. For example, as we discuss in Chapters 4, a different but similar heuristic was applied in our flexible scheduler for workflow ensembles.

3.7 Results

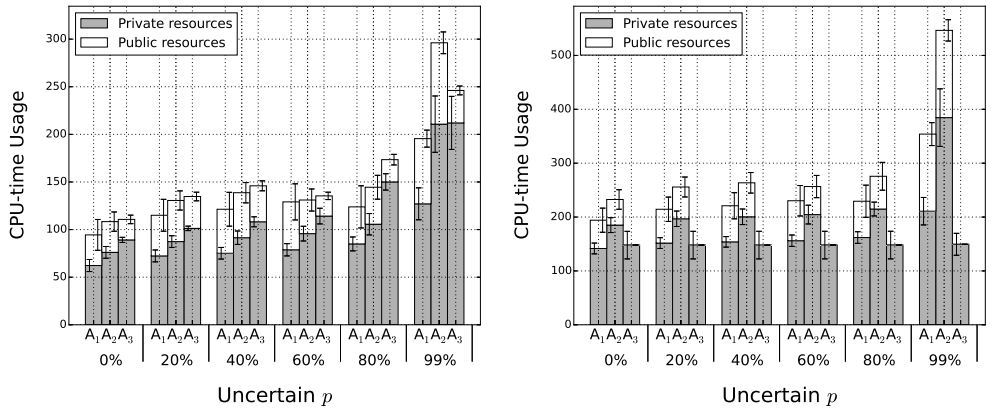
This section presents numerical results that were produced to assess the impact of imprecise information on the available bandwidth in inter-cloud links as well as the effectiveness of using a constant value and a value derived by the proposed procedure for the deflating factor that is applied to the available bandwidth.

Algorithm 7 First step of the experimental evaluation – Database creation

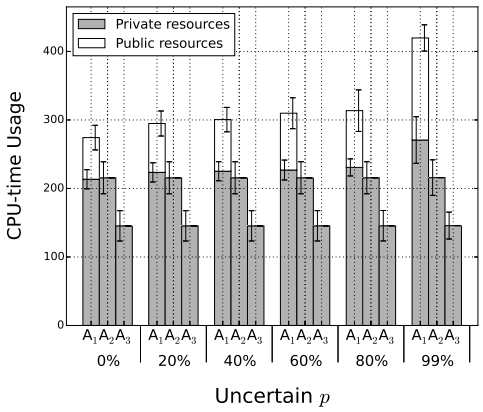
1: $A_{pps} = \{Montage, LIGO, CyberShake\}$
2: $S_{izes} = \{50, 100, 200, 300\}$ ▷ Number of tasks
3: $S_{chedulers} = \{HCOC, PSO_{original}, PSO_{adapted}\}$
4: $D_{eflating\_factors} = \{0, 5, 15, 25, 35, 45, 50\}$ ▷ Fixed deflating values in percentagem
5: $B_{andwidths} = \{10, 20, 30, 40, 50, 60\}$ ▷ In Mbps
6: $V_{ariability} = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99\}$ ▷ In percentagem
7: $D_{eadlines} = \{\frac{2\times T_{max}}{7}, \frac{3\times T_{max}}{7}, \frac{4\times T_{max}}{7}, \frac{5\times T_{max}}{7}, \frac{6\times T_{max}}{7}\}$
8: **for each** $\mathcal{S}$ in $S_{chedulers}$ **do**
9: **if** $\mathcal{S}$== "$PSO_{original}$" **then**
10: $D_{eflating\_factors} = \{+\infty\}$
11: **end if**
12: **for each** *app* in A_{pps} **do**
13: **for each** *size* in S_{izes} **do**
14: Create the database $\mathcal{H}_{\mathcal{G},\mathcal{S}}$
15: $\mathcal{G}$ = DAG that represents the application *app* with the number of tasks of *size*
16: Create 100 DAGs of $\mathcal{G}$ differing only in the weights of nodes and edges
17: **for** $i = 1$ **to** 100 **do**
18: **for each** b in $B_{andwidths}$ **do**
19: **for each** $\mathcal{U}$ in $D_{eflating\_factors}$ **do**
20: **for each** d in $D_{eadlines}$ **do**
21: Generate a schedule for the DAG $\mathcal{G}_i$ with a deadline value of d by using the scheduler $\mathcal{S}$ and an available bandwidth of b Mbps deflated by $\mathcal{U}$%, e.g, $b = b\,(1 - \frac{u}{100})$
22: **for each** p in $V_{ariability}$ **do**
23: Simulate the execution of the schedule given by step 21 and calulate the makespan m and cost c of the workflow by assuming b with a variability of $\pm p$%
24: $\mathcal{H}_{\mathcal{G},\mathcal{S}} = \mathcal{H}_{\mathcal{G},\mathcal{S}} \cup \langle b, \mathcal{U}, d, p, m, c\rangle$ ▷ Store the 6-tuple in the database
25: **end for**
26: **end for**
27: **end for**
28: **end for**
29: **end for**
30: Store the database $\mathcal{H}_{\mathcal{G},\mathcal{S}}$
31: **end for**
32: **end for**
33: **end for**

3.7.1 Impact of Imprecise Information about the Available Bandwidth on Inter-Cloud Links

This section assess the variability of the bandwidth estimates on the produced schedules. This section evaluates how the use of imprecise estimates can increase the CPU usage as well as the failure of the target objectives for the scheduling problem. In this section, for each schedule solution given by the scheduler that satisfies the deadline (qualified solution), the estimates of cost and makespan were re-calculate as the uncertainty (variability) p of the estimated available bandwidth b increases. As the purpose of this evaluation is to assess the impact of the non-expected variability p of the bandwidth estimates, since schedulers assume that b is a precise information and does not undergo any kind of variability, the proposed procedure was not applied.



(a) $\mathcal{D}_{\mathcal{G}} = \mathcal{T}_{max} \times 2/7$ (b) $\mathcal{D}_{\mathcal{G}} = \mathcal{T}_{max} \times 4/7$



(c) $\mathcal{D}_{\mathcal{G}} = \mathcal{T}_{max} \times 6/7$

Figure 3.11: Averages of CPU usage of private and public resources of the HCOC scheduler for scheduling the CyberShake application of 50 tasks when the measured inter-cloud available bandwidth of 30 Mbps was varied from 0% to 99% for all private cloud sizes A_1, A_2 and A_3 of the Table 3.2.

a) Resource demand

The goal of this evaluation is to assess for different bandwidth availability in the inter-cloud link the needs of leasing public resources to meet execution deadline of workflows. The CPU-time usage for the execution of workflows was evaluated considering three private clouds: A_1, A_2 and A_3 (Table 3.2). Workflow templates with 50, 100, 200 and 300 tasks for three different applications were employed in the evaluation. For each workflow template, twenty workflows were created with different values of the weight of nodes and edges, for a total of 240 workflows. Each workflow was simulated using 3 different scheduling algorithms, 5 different deadline values, 6 inter-cloud estimates of available bandwidth and 11 different values of p. Therefore, results were derived from an extensive set of $712,800$ simulations.

Figure 3.11 shows the averages CPU usage in the private, as well as in the public cloud for the HCOC scheduler to schedule a workflow with 50 tasks of the CyberShake application when the measured inter-cloud available bandwidth of a 30 Mbps link was varied from 0% to 99%, with this usage shown as a function of the size of the private

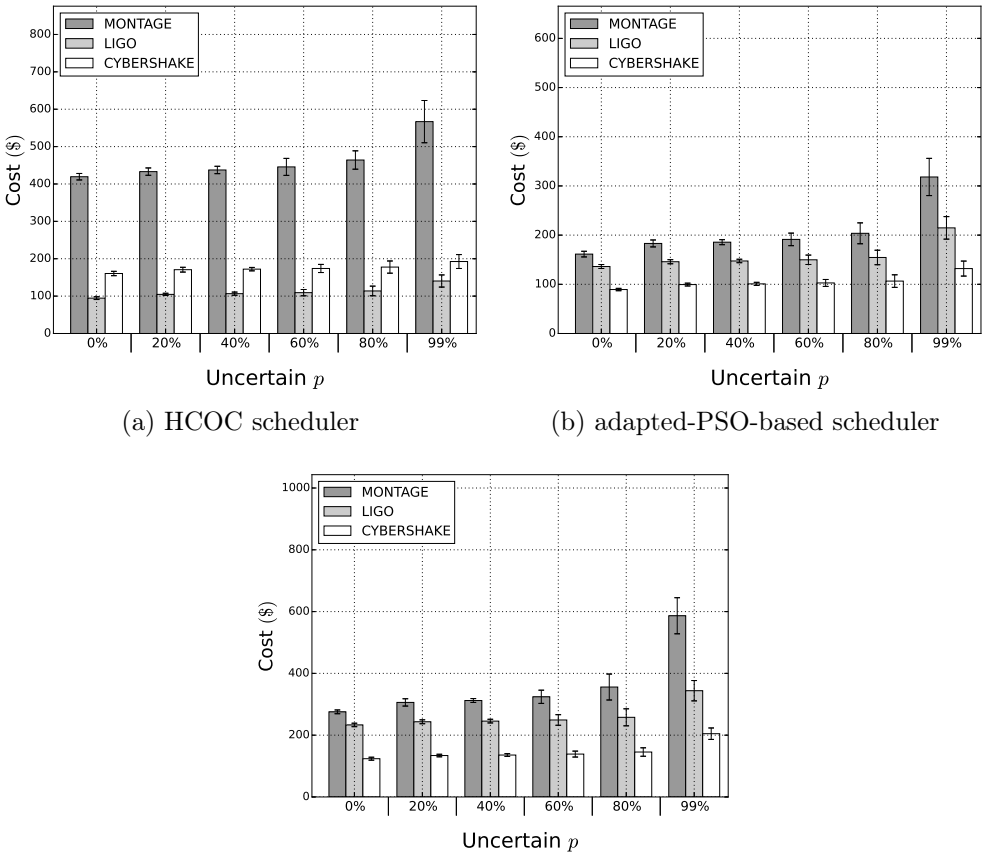
cloud and the p value. Figure 3.11 evinces the increase in CPU-time usage with increasing uncertainty in the available bandwidth estimates (for a better visualisation, the results for p values of 10%, 30%, 50%, 70% and 90% are not plotted). The increase in data transfer times implies longer periods of CPU usage because the jobs need to wait longer for the data they depend on. This increase in CPU-time usage increases the idle periods during which the CPU is allocated and becomes unavailable for processing other dependent jobs, which can lead to the need to lease resources from the public cloud. Figure 3.11 shows the increase in CPU-time usage triggered by the uncertainty in the available bandwidth estimates.

Figure 3.11 shows results for three different deadline values: a low, a median and a high value, each one expressed as a proportion of the $\mathcal{T}_{max}$ value. In most cases, the low deadline can not be met when using the private cloud A_3, and resources from the public cloud must consequently be leased (Figure 3.11a). However, for a median deadline, resources in the private cloud A_3 were generally sufficient, which did not happen when the workflows are executed in private cloud A_1 and A_2 (Figure 3.11b). For a high deadline value, execution only on the private cloud A_1 was not satisfactory (Figure 3.11c). In general, private clouds A_2 and A_3 have either sufficient or nearly sufficient resources to process the required workload. Because the aim of this study is to analyse the impact of the uncertainty of the available bandwidth in the inter-cloud links, results will be shown only for the private cloud A_1, since its resources were not sufficient to meet the application deadline, independent of the workflow application, scheduling algorithm, deadline value, and available bandwidth in the inter-cloud links.

b) Failure to Achieve the Established Objective

Various simulations were performed to assess the impact on the achievement of the objectives expressed in the objective function of the algorithms. This evaluation involved $1,200$ workflows (3 types of application workflow templates $\times$ 4 different sizes per template $\times$ 100 workflows per application and size). These workflows were simulated by using 6 different available bandwidth values, 11 p values and 5 deadline values. The evaluation summarises the outcome of an extensive set of $79,200$ simulations for the PSO-based scheduler and $396,000$ simulations for the HCOC and adapted-PSO-based schedulers. The number of simulations using the PSO-based scheduler was five times lower since the deadline value is not used as part of the input data. Figure 3.12 shows the cost estimates given by the HCOC, adapted-PSO-based and PSO-based schedulers in scheduling a 50-task workflow application when the measured inter-cloud available bandwidth of a 40 Mbps link was between 0% and 99%.

The Montage and CyberShake applications have a similar structural pattern in which the DAG jobs send the same intermediate file to two or more successor jobs. Moreover, depending on how the schedule was produced, sending multiple copies of the intermediate file can increase the traffic in the inter-cloud links [87]. Because the PSO-based scheduling algorithm does not consider the available bandwidth between resources but only the cost of data transmission from/to the public cloud, unnecessary flooding of the inter-cloud links can occur. This implies an increase of the execution cost of the Montage and CyberShake



(a) HCOC scheduler

(b) adapted-PSO-based scheduler

(c) PSO-based scheduler

Figure 3.12: Barplots of the cost estimates given by the HCOC, adapted-PSO and PSO-based schedulers to schedule 50-task workflow application when the error in the estimated available bandwidth of 40 Mbps varies from 0% to 99%. For HCOC and adapted-PSO-based schedulers, the deadline value was $\mathcal{D}_{\mathcal{G}} = \mathcal{T}_{max} \times 4/7$.

applications (Figure 3.12c). For example, when applying an uncertainty value of 20% to the inter-cloud available bandwidth of a 40 Mbps that was used to produce the original schedule, the average cost of running the Montage application in hybrid clouds increased by 12%, while for an uncertainty value of 99%, the cost doubled. For CyberShake, the cost increased by 9% for a p value of 20%, while for a p value of 99% the cost doubled as well. Although the LIGO application sends fewer duplicate intermediate files to other dependent jobs, which contributes with less traffic in the inter-cloud links, the uncertainty in the available bandwidth estimates cooperated with the increase in the cost estimates. The cost increased 4% for $p = 20\%$ and only 47% for $p = 99\%$.

For each qualified schedule, a variability $p > 0$ in the available bandwidth estimates was introduced by the simulator that checks if the schedule remained qualified. Table 3.3 shows the percentage of solutions that missed the deadline (disqualified solutions) for the HCOC and adapted-PSO-based schedulers. Schedules produced by the PSO-based scheduler were not classified as either qualified and disqualified since it is a cost-aware scheduler only.

Because both HCOC and the adapted-PSO-based are deadline- and cost-aware schedulers, they are expected to produce schedules that meet the deadline with low costs.

Table 3.3: Percentagem of disqualified solutions of Figure 3.12

		Uncertain p%					
		0	20	40	60	80	99
HCOC	Montage	12%	58%	64%	82%	91%	92%
	LIGO	42%	47%	49%	56%	62%	79%
	CyberShake	1%	22%	24%	26%	28%	63%
Adapted-PSO-based	Montage	0%	16%	17%	18%	39%	95%
	LIGO	1%	20%	20%	25%	24%	74%
	CyberShake	3%	19%	21%	19%	23%	49%

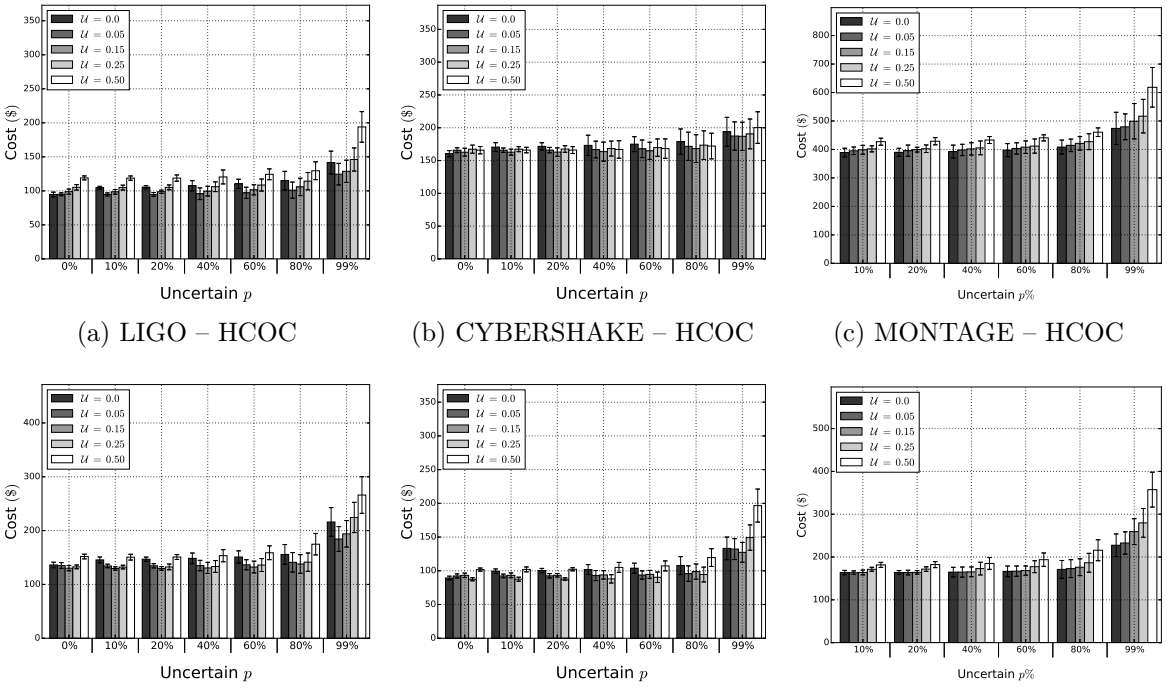
However, the effect of errors in the inter-cloud available bandwidth estimates resulted in increased makespans, which caused the deadline to be missed. For example, of the 88% of the solutions for Montage produced by HCOC that were eligible when $p = 0\%$, 58% were disqualified for $p = 20\%$, and 91% were disqualified for $p = 80\%$. The schedules produced by adapted-PSO were less impacted. For example, of the 100% of the Montage solutions that were eligible when $p = 0\%$, 16% were disqualified when $p = 20\%$, and 39% were disqualified when $p = 80\%$. The cost estimates provided by HCOC, increased by 5% when the p value varied from 0% to 20%, and 30% when p value varied from 0% to 80%. The cost estimates provided by the adapted-PSO-based also increased with increasing values of p. The cost estimate increased by 14% when the p value varied from 0% to 20%, and 28% when p value varied from 0% to 80%. Similar trends were obtained for the LIGO and CyberShake workflows; i.e, increasing the error in the estimated available bandwidth, increases the number of disqualified schedules as well as underestimates costs.

This section showed results for schedulers that were not designed to address with inaccurate information about the available bandwidth and produced schedules with misleading cost and makespan estimates. The next section will show that the use of a deflating factor for the available bandwidth that is used as an input to the scheduler can reduce the disqualification of schedules, by avoiding underestimates of the makespan and cost. Because the original PSO-based scheduler does not consider the available bandwidth to produce schedules, it was not evaluated hereafter.

3.7.2 Using a Fixed Deflating $\mathcal{U}$ value

Deflating the measured available bandwidth value using the $\mathcal{U}$ value and providing the deflated bandwidth estimate to the scheduler at the scheduling time will hopefully reduce the negative impact of imprecise bandwidth information on the production of schedules. The results from the next experiments assess the extent to which a deflating factor $\mathcal{U}$ mitigates the negative impact of imprecise information about the available bandwidth on the produced schedules.

One thousand two hundred workflows were employed in this evaluation (3 types of application workflow template $\times$ 4 different workflow size per template $\times$ 100 workflows per application and size). These workflows were used in simulations with 6 different available bandwidth values, 11 p values, 5 deadline values and 6 deflating $\mathcal{U}$ values. The experiment used a fixed $\mathcal{U}$ value, $\mathcal{U} \in \{0, 5, 15, 25, 35, 45, 50\}$, but for better visualization, results for $\mathcal{U} = 35$ and $\mathcal{U} = 45$ were not plotted. This experiment summarises the



(a) LIGO – HCOC (b) CYBERSHAKE – HCOC (c) MONTAGE – HCOC

(d) LIGO – Adapted-PSO (e) CYBERSHAKE – Adapted-PSO (f) MONTAGE – Adapted-PSO

Figure 3.13: Barplots of the costs of disqualified solutions when the measured available bandwidth of 40 Mbps was deflated by a fixed $\mathcal{U}$ value and provided as input to the HCOC (top) and adpted-PSO-based (bottom) schedulers to schedule the applications workflows of 50 tasks with the deadline value equal to $\mathcal{D}_{\mathcal{G}} = \mathcal{T}_{max} \times 4/7$. The x-axis represents the estimated error in the available bandwidth which varies from 10% to 99%.

outcome of a set of $2,376,000$ simulations for each deadline- and cost-aware scheduler.

Figure 3.13 shows the cost estimates given by the HCOC and adapted-PSO-based schedulers to schedule a 50-task workflow application when the uncertainty of the measured inter-cloud available bandwidth of 40 Mbps and 60 Mbps varied from 0% to 99%. Table 3.4 shows the percentage of disqualified solutions shown in Figure 3.13 when the available bandwidth estimates were deflated by a fixed $\mathcal{U}$ value.

The results show a reduction in the percentage of disqualified schedules when the deflating $\mathcal{U}$ value was employed. This trend was present for most cases with the three types of applications. For example, using HCOC with $\mathcal{U} = 0.05$ and $p = 60\%$, there was an increase of 10%, 21% and 22% in the number of qualified solutions for the LIGO, CyberShake and Montage applications, respectively; while for the adapted-PSO, the number of qualified solutions increased 21%, 20% and 8% for LIGO, CyberShake and Montage application, respectively.

However, the use of a large deflating $\mathcal{U}$ value increased the number of disqualified solutions. For HCOC, when $\mathcal{U} = 0.5$ and $p = 60\%$, the number of disqualified solutions for LIGO, CyberShake and Montage increased by 15%, 5% and 13%, respectively; while using the adapted-PSO caused increases of 15%, 20% and 24% for LIGO, CyberShake and Montage, respectively. This experiment also suggests that the number of disqualified solutions can be reduced in the presence of high variability of the available bandwidth. For example, for an uncertainty value of 99% and $\mathcal{U} = 0.05$, the numbers of qualified solutions produced by HCOC increased by 23% for CyberShake and 3% for LIGO. For the adapted-PSO-based, the increases were 2% for Montage, 21% for LIGO, and 7% for CyberShake. In general, regardless of the scheduler and workflow application, the act of deflating the available bandwidth estimates contributed to the increase the number of schedules that satisfy the deadline even when the bandwidth estimates underwent to high variabilities.

Table 3.4: Percentages of disqualified solutions of the evaluation for the HCOC scheduler (left) and the Adapted-PSO (right) of Figure 3.13

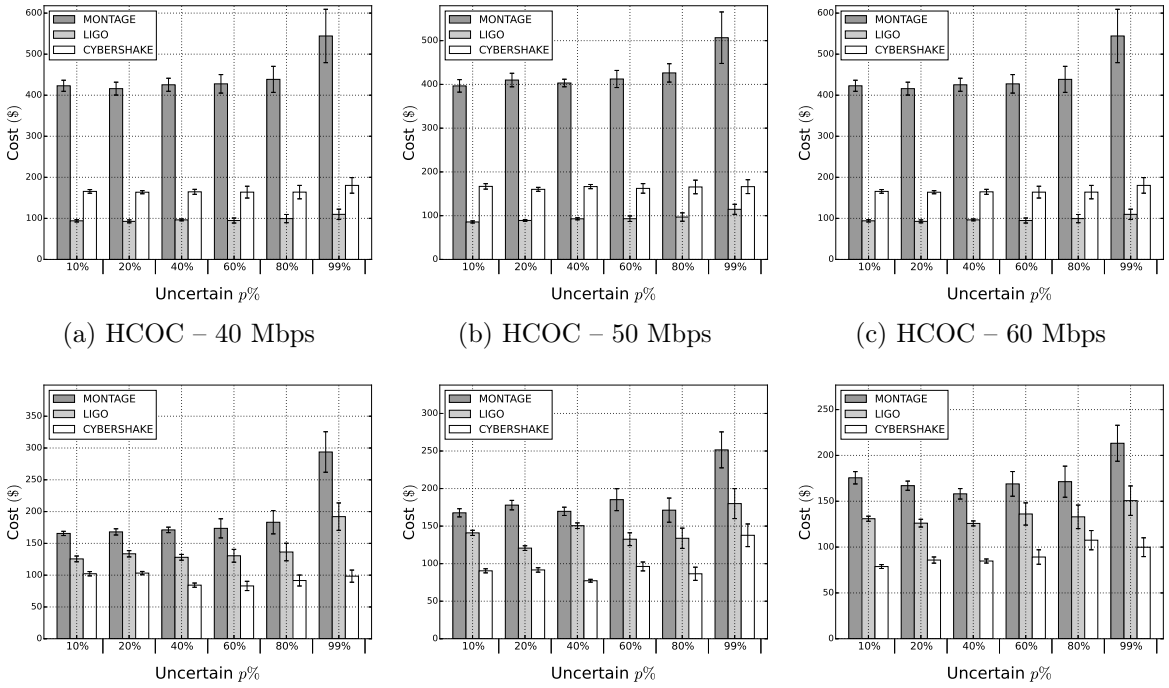
HCOC		Uncertain p%					
	$\mathcal{U}$ **value**	10	20	40	60	80	99
LIGO	0.0	52%	55%	61%	67%	81%	85%
	0.05	43%	53%	53%	57%	56%	66%
	0.15	45%	46%	47%	59%	62%	70%
	0.25	46%	47%	47%	60%	58%	75%
	0.50	79%	80%	80%	82%	86%	88%
CyberShake	0.0	16%	17%	22%	23%	31%	80%
	0.05	2%	2%	2%	2%	5%	13%
	0.15	1%	1%	1%	1%	6%	24%
	0.25	6%	6%	6%	6%	10%	22%
	0.50	22%	24%	26%	28%	37%	57%
Montage	0.0	13%	58%	64%	82%	91%	92%
	0.05	1%	49%	52%	60%	79%	85%
	0.15	1%	74%	73%	75%	81%	87%
	0.25	2%	77%	79%	83%	87%	89%
	0.50	74%	90%	92%	95%	95%	95%

Adapted-PSO		Uncertain p%					
	$\mathcal{U}$ **value**	10	20	40	60	80	99
LIGO	0.0	18%	19%	17%	22%	22%	61%
	0.05	3%	3%	3%	3%	3%	6%
	0.15	11%	11%	11%	11%	12%	13%
	0.25	14%	14%	14%	14%	16%	19%
	0.50	20%	20%	20%	32%	35%	40%
CyberShake	0.0	18%	17%	18%	25%	24%	52%
	0.05	5%	5%	5%	5%	7%	8%
	0.15	10%	10%	10%	11%	11%	13%
	0.25	14%	14%	14%	14%	15%	19%
	0.50	22%	23%	24%	40%	40%	45%
Montage	0.0	14%	15%	17%	13%	14%	67%
	0.05	2%	4%	4%	5%	7%	12%
	0.15	12%	13%	13%	15%	17%	19%
	0.25	14%	19%	19%	22%	26%	23%
	0.50	21%	33%	33%	37%	53%	65%

The use of a non-null deflating factor $\mathcal{U}$ did not imply an increase in estimated costs, as shown in Figure 3.13. For example, using HCOC to schedule the LIGO application

when $p = 40\%$ led to costs 13% lower for $\mathcal{U} = 0.05$ instead of $\mathcal{U} = 0$. In other cases, the use of a large deflating $\mathcal{U}$ value resulted in a further reduction in costs instead of using a short $\mathcal{U}$ value. For example, when the adapted-PSO algorithm was used to schedule CyberShake application for $p = 10\%$, the cost was 6% lower when using $\mathcal{U} = 0.25$ instead of $\mathcal{U} = 0.05$. The decrease in cost was possible since the set of virtual machines selected by the algorithm to perform the same task scheduling is different when the available bandwidth estimates are deflated by different $\mathcal{U}$ values.

These experiments suggest that the decrease in the number of disqualified solutions depends on the $\mathcal{U}$ value and that there is no clear trend for choosing an optimum value. This motivated the adoption of a procedure that considers the execution history to determine the most appropriate $\mathcal{U}$ value.



(a) HCOC – 40 Mbps (b) HCOC – 50 Mbps (c) HCOC – 60 Mbps

(d) adapted-PSO-based – 40 Mbps (e) adapted-PSO-based – 50 Mbps (f) adapted-PSO-based – 60 Mbps

Figure 3.14: Barplots of the costs when the measured available bandwidths 40 Mbps, 50 Mbps and 60 Mbps values deflated by a specific $\mathcal{U}$ value and provided as input to the HCOC (top) and adapted-PSO-based (bottom) schedulers to schedule the applications workflows of 50 tasks with the deadline value equal to $\mathcal{D}_{\mathcal{G}} = \mathcal{T}_{max} \times 4/7$.

3.7.3 Using the Proposed procedure to Calculate the $\mathcal{U}$ value

When the $\mathcal{U}$ value is derived using the proposed procedure, qualified schedules were produced for almost all p values. The analysis of the execution history is the key to determining a specific deflating factor for the $\mathcal{U}$ value. The results show the extent to which using the $\mathcal{U}$ value that is computed considering the execution history can mitigate the negative impact of imprecise information about the available bandwidth. This section shows the results of simulations when the employment of the proposed procedure to calculate a deflating factor $\mathcal{U}$ to the measured available bandwidth value was used as an

input to the scheduler. This experiment summarizes a set of 396,000 simulations for each deadline- and cost-aware scheduler (3 application workflow templates × 4 different sizes × 100 workflows per size × 6 available bandwidth values × 11 p values × 5 deadline values).

A comparison of the results obtained when the proposed MLR procedure was not employed (Section 3.7.1) showed a decrease in the percentage of disqualified schedules. This is true independent of the p value for the three types of applications and for both schedulers. For example, by comparing Table 3.3 with 3.5 using the employment of the HCOC scheduler for a measured available bandwidth of 40 Mbps was deflated by a specific $\mathcal{U}$ value with $p = 20\%$, the number of qualified solutions increased by 10%, 22% and 51% for the LIGO, CyberShake and Montage applications, respectively. When the adapted-PSO-based was employed, the number of qualified solutions increased by 17%, 16% and 13% for the LIGO, CyberShake and Montage applications, respectively. Even in scenarios with high uncertainties in the available bandwidth estimates, a noticeable increase in the number of qualified solutions was found. For example, using HCOC and $p = 99\%$, the number of qualified solutions for the LIGO, CyberShake and Montage applications increased by 35%, 35% and 28%, respectively; while for the adapted-PSO-based this increased by 31%, 19% and 12%, respectively.

Table 3.5: % of disqualified solutions of the evaluation for the HCOC scheduler (left) and the Adapted-PSO (right) of Figure 3.14

HCOC		Uncertain p%					
		0	20	40	60	80	99
40 Mbps	LIGO	33%	37%	40%	47%	60%	70%
	CyberShake	0%	0%	3%	7%	7%	50%
	Montage	3%	7%	23%	27%	77%	90%
50 Mbps	LIGO	30%	20%	23%	37%	43%	70%
	CyberShake	0%	0%	3%	0%	7%	40%
	Montage	7%	0%	3%	7%	33%	87%
60 Mbps	LIGO	3%	3%	3%	0%	7%	43%
	CyberShake	3%	10%	0%	3%	3%	3%
	MONTAGE	7%	3%	3%	13%	20%	83%

Adapted-PSO		Uncertain p%					
		0	20	40	60	80	99
40 Mbps	LIGO	3%	3%	3%	0%	7%	43%
	CyberShake	3%	3%	0%	3%	3%	30%
	Montage	7%	3%	3%	13%	20%	83%
50 Mbps	LIGO	30%	20%	23%	37%	43%	70%
	CyberShake	0%	3%	0%	0%	0%	20%
	Montage	7%	0%	3%	7%	33%	87%
60 Mbps	LIGO	0%	0%	0%	0%	3%	13%
	CyberShake	0%	0%	0%	3%	3%	7%
	Montage	7%	7%	3%	7%	10%	53%

The use of a specific $\mathcal{U}$ value not only increases the number of qualified solutions, but also achieves slightly less expensive schedules, as shown in Figure 3.14. For example, using HCOC and $p = 40\%$, the costs of the LIGO, CyberShake and Montage applications were respectively 9%, 5% and 4% lower than the original cost estimates (Figures 3.13 with $\mathcal{U} = 0$). The adapted-PSO-based scheduler reduced the costs of these LIGO, CyberShake and Montage applications by 5%, 4% and 2%, respectively. The results show that when the deflated values for the measured available bandwidths are used, the scheduler produces schedules with higher utilisation of resources, thus ensuring that the schedules meet the deadlines as well as reducing misleading cost estimates.

3.7.4 Discussion

Most deterministic schedulers in the literature neglect the presence of variability in the bandwidth estimates in their design. To solve this issue, this section provides a procedure for such schedulers in order to enhance their robustness under uncertainties of input values,

especially bandwidth availability. As this section shows, deterministic schedulers may make decisions (produce schedules) that lead to ineffective executions of applications using imprecise information regarding the available bandwidth. We argue that the negative impact of using imprecise values may be reduced thought the adoption of mechanisms that take into consideration the uncertainties of such input values.

This section presents a proactive approach (procedure) that refines the bandwidth estimates given by monitoring tools *before* furnishing it to the scheduler. The results shown in this section indicate that the employment of the proposed procedure was able to reduce the impact of using misleading bandwidth estimates and increase the number of qualified solutions. The proposed procedure produces a deflating multiplier value that is applied to the measured available bandwidth, which is given by such tools, before being furnished as an input to the scheduler. According to an expected uncertainty p of the available bandwidth measurement, the procedure deflates the measured bandwidth to prevent misleading information from affecting the performance of the scheduler in producing poor schedules. The results presented in this section evidence the production of robust schedules using ordinary deterministic schedulers when our procedure was employed. The results have indicated the performance of these schedulers fed by imprecise information underwent significant degradation than the performance of the same schedulers when our procedure was used. Also, the use of the proposed procedure did not imply in expensive schedules by deterministic scheduler, which implies that a higher utilisation of resources in the private cloud was achieved. It is worth mentioning that the proposed procedure does *not* modify the original algorithm of the schedulers, it is just a proactive procedure that assists them in producing robust schedules under imprecise bandwidth estimates. The proposed procedure comes into play only for applying a deflation on the bandwidth estimates given by the monitoring tools at scheduling time, since such original estimates may undergo high variabilities that may impacts the schedules performance. Once again, the use of a deflated value of bandwidth estimates to produce schedules is an attempt to increase the robustness of such schedules under the uncertainty of the bandwidh estimates measured by such tools.

A $\mathcal{U}$ value computed by the proposed procedure is dependent on several parameters, such as the type of workflow, the deadline value, the value of the measured available intra- and inter-cloud bandwidth, the error occurred in previous estimates of the makespan and the cost due to the presence of an uncertainty value on the bandwidth value. Because several parameters can influence the $\mathcal{U}$ value, applying a fixed value as in the previous sub-section does not produce a single best deflating factor value, because it can either underestimate or overestimate the available bandwidth, which leads to a deadline violation or unnecessary rental of processing power on public clouds. Thus, for a particular workflow application, a deadline value, an available bandwidth value, an uncertainty value, and a scheduler, the execution history assists the proposed procedure to determinate a specific $\mathcal{U}$ value that tries to avoid the disqualification of the schedule.

Note that the computation of the $\mathcal{U}$ value depends on the size of the execution history. The greater the number of tuples in the $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ subset, the greater the amount of information used by the multiple linear regression to yield the linear equation $\mathcal{U} = f(x, y)$. For example, in our work in [67], the multiple linear regression was obtained by using 25%,

50%, 75% and 100% of the historical data. When using half of the execution history, better results were obtained when a quarter of this historical data was used. A simulation pattern of improvement was also found when three quarters of the historical data were used in the regression instead of half of it. However, we noticed that there was little improvement in the results when the amount of historical data considered increased from 75% to 100%. Therefore, in the presented evaluation of the proposed procedure, it was considered only the employment of 100% of the historical data.

3.8 IPDT-Mechanism *vs.* IP-FULL-FUZZY

This section introduces two schedulers: the IPDT [17, 19] and the IP-FULL-FUZZY [16, 19]. Both schedulers treat the workflow scheduling problem as an integer linear programming[13] (ILP) problem to produce schedules. The main difference among IPDT and IP-FULL-FUZZY is that the IP-FULL-FUZZY takes into account uncertainties of applications demands as well as of resource availability, while the IPDT does not do this. In other words, similar to the previous HCOC and PSO-based scheduler, the IPDT is an ordinary deterministic scheduler that was not designed to deal with imprecise information of input data, while the IP-FULL-FUZZY is a robust scheduler under uncertainties of input data. In fact, the IPDT was employed for the design of the IP-FULL-FUZZY scheduler, which models such uncertainties as fuzzy numbers. This section focuses, therefore, in the comparison between the IPDT using the procedure (hereinafter called as the IPDT-Mechanism) and the IP-FULL-FUZZY.

To assess the effectiveness of the proposed procedure, the schedule produced by the IP-FULL-FUZZY was compared via simulation to the one produced by the IPDT using the proposed procedure (mechanism). Simulation results show that the IPDT-Mechanism underwent less degradation in its performance than did the IP-FULL-FUZZY scheduler as the degree of uncertainty in the available bandwidth increases. Results evidenced that the ability of the proposed procedure in enhancing the robustness of the IPDT (IPDT-Mechanism) was better than that of the IP-FULL-FUZZY that employs fuzzy number for this purpose.

This section is organised as follows. An overview of the IPDT scheduler is presented in Section 3.8.1, while the description of the IP-FULL-FUZZY is described in Section 3.8.2. The evaluation methodology of how the proposed procedure was employed in the IPDT and compared to the IP-FULL-FUZZY is elaborated in Section 3.8.3. Results are discussed in Section 3.8.4, while final remarks are presented in Section 3.8.5.

3.8.1 An Overview of the IPDT Scheduler

The deterministic scheduler Integer Programming with Discrete Time (IPDT) [17] was designed to schedule workflow applications on grids with the objective of minimising the execution time (*makespan*) of the application. The scheduling process is based on

[13] Apart from heuristics and metaheuristics, mathematical programming such as linear or mixed-integer programming is often used as a tool to solve workflow scheduling problems [69, 71, 72]. For example, the HCOC scheduler is developed using a heuristic [35], while the PSO-based scheduler a meta-heuristic [104].

random search driven by an integer programming problem. The IPDT scheduler accepts two graphs as input. The graph $H = (V_H, E_H)$ represents the resource topology, while the DAG $D = (V_D, A_D)$ indicates the dependencies among tasks. In H, V_H is the set of m ($m = |V_H|$) resources connected by the set of links E_H. Resources are labeled from 1 to m. In D, V_D is the set of n ($n = |V_D|$) tasks with integer numbers as labels, which allow a topological ordering of tasks (since it is a DAG), and A_D is the set of dependencies.

The IPDT scheduler considers that the input DAG has a single input task and a single output task. DAGs failing to satisfy this condition because they involve more than one input or output task can be easily modified by considering two *null tasks* with both processing time and communication weight equal to zero.

The IPDT scheduler provides as output a list, which provides information about the resource on which each task should be executed, the starting time of that task, and the time when data transfer should take place. Some characteristics of the DAGs include:

- I_i: the processing demand of the ith task of V_D, expressed as number of instructions to be processed by the ith task ($I_i \in \mathbb{R}_+$ and $i \in V_D$); and
- $B_{i,j}$: the number of bytes transmitted between the ith task and the jth task ($B_{i,j} \in \mathbb{R}_+$ and $i, j \in \mathbb{R}_+$);

The weights of arcs (B) and nodes (I) representing, respectively, the amount of data to be transferred and the amount of processing of the application represented by the DAG D, which is furnished by the user. Some characteristics of the environment execution include:

- TI_k: the time that the kth resource takes to execute 1 instruction ($TI_k \in \mathbb{R}_+$ and $i \in V_H$);
- $TB_{k,l}$: the time for transmitting 1 bit on the link connecting the kth resource to the lth resource ($TB_{k,l} \in \mathbb{R}_+$ and $k, l \in V_H$). The $TB_{k,l}$ value represents the available bandwidth on the path between k and l;
- $\delta(k)$: the set of resources linked to the kth host in the network, including the resource k itself; and
- $\tau(k)$: the number of processing cores of the resource k. As the original IPDT scheduler only accepts single-core resources in its ILP formulation, a small adaption was carried to accept multi-core resources as well.

The schedule achieved by the IPDT scheduler is given by the values of variables $x_{i,t,k} \in \{0, 1\}$; $x_{i,t,k}$ is a binary variable that assumes a value of 1 if the ith task finishes at time t on the resource k; otherwise this variable assumes a value of 0. The objective of the scheduler is to minimise the execution time of the application, which coincides with the execution time of the last task of the DAG, the nth task. For convenience, the ILP formulation of the scheduling problem solved by the IPDT uses the notation $\mathcal{T} = \{1, \ldots, T_{max}\}$ to represent the discrete timeline, where T_{max} is the time that the application would take to run all the tasks serially on the fastest resource of V_H.

The T_{max} value is defined as follows:

$$T_{max} = \min_{\forall k \in V_H} \left\{ TI_k \right\} \times \sum_{i=1}^{n} I_i \tag{3.1}$$

The IPDT scheduler is given by the following integer programming problem:

$$\text{Minimise} \sum_{t \in \mathcal{T}} \sum_{k \in V_H} t \times x_{n,t,k}$$

Subject to:

$$\sum_{t \in \mathcal{T}} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{for } j \in V_D; \qquad (C1)$$

$$x_{j,t,k} = 0 \qquad \text{for } j \in V_D, \quad k \in V_H \quad t \in \{1, \ldots, \lceil I_j TI_k \rceil\}; \qquad (C2)$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - I_j TI_l - B_{i,j} TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \quad \text{for } j \in V_D, \quad ij \in A_D, \quad l \in V_H, \quad t \in \mathcal{T}; \qquad (C3)$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + I_j TI_k - 1 \rceil} x_{j,s,k} \leq \tau(k) \qquad \text{for } k \in V_H, \quad t \in \mathcal{T}, \quad t \leq \lceil T_{max} - I_j TI_k \rceil; \qquad (C4)$$

$$x_{j,t,k} \in \{0, 1\} \qquad \text{for } j \in V_D, \quad l \in V_H \quad t \in \mathcal{T} \qquad (C5)$$

The constraints in (C1) determines that a task must be executed in a single resource only, while (C6) defines the integer domain for variables $x_{j,t,k}$ in the formulation. The constraint in (C2) determines that a task j cannot terminate until all its instructions have been completely executed, while the constraints in (C3) establish that the jth task cannot start execution until all j's predecessors have finished their execution and transferred the data required by the jth task. Lastly, the constraint in (C4)[14] specifies that the number of tasks executing on resource k at a given time s cannot exceed the number of processing cores of k.

The IPDT scheduler does not consider uncertainties of both application demands and of resource availability. According to the authors [17], *uncertainties of application demands arise from the lack of precision in estimating the amount of data to be transferred by different* [workflow] *application* [descriptions], *while uncertainty of available bandwidth*

[14] In the original ILP formulation of the IPDT scheduler in [17], the constraints in (C4) was defined as *there is at most a single task in execution on any one resource at a specific time*. This constraint was modified to accept multi-core resources.

is related to the nature of measurement and monitoring tools. To develop a workflow scheduler aware of uncertainties of application demands and resource availability, the IPDT scheduler was enhanced to take into account such uncertainties through the employment of fuzzy numbers in its formulation. This new version of the IPDT scheduler, called IP-FULL-FUZY, is described in the next section.

3.8.2 An Overview of the IP-FULL-FUZZY Scheduler

The IPDT was employed for the design of the IP-FULL-FUZZY scheduler [16], which models the uncertainties of application demands and resource availability as fuzzy numbers. The IP-FULL-FUZZY scheduler is based on a fuzzy optimisation formulation. According to the authors, this formulation is then transformed into a crisp equivalent model based on an integer programming formulation. In other words, to solve the scheduling problem, the fuzzy constraints and the objective function must be transformed into corresponding crisp expressions, leading to an integer programming formulation of the original problem. For convenience, this section presents only the crisp equivalent model and for more details about the fuzzy formulation itself, please see [16]. Importantly, the uncertainty level existing in the input data describing the workflow application or the resources is assumed as *actual uncertainty level*, and the uncertainty level which the IP-FULL-FUZZY scheduler was designed to handle is considered as *projected uncertainty level*.

Similar to the IPDT scheduler, the IP-FULL-FUZZY scheduler implements an integer linear program, which is the result of the mapping of the fuzzy formulation onto a crisp formulation. The IP-FULL-FUZZY scheduler also solves the scheduling problem through an integer linear program, which attempts to minimise the makespan of the workflow application represented by the DAG. The IP-FULL-FUZZY scheduler also accepts two graphs as input. The graph $H = (V_H, E_H)$ that represents the resource topology indicates the availability of computation and communication resources of the execution environment. The DAG $D = (V_D, A_D)$ indicates the dependencies among tasks and represents the computational and the communication demands of the workflow application. To take into account the uncertainty of application demands and resource availability, triangular fuzzy numbers was employed on the scheduling modelling. Triangular fuzzy numbers are often used to represent mathematically uncertain quantities that do not have a more specific membership function (or degree of truth) to describe them. The value of a fuzzy number varies in an interval $[min, max]$ and is associated with a membership function $\mu(x)$, $x \in \mathbb{R}$, $\mu(x) \in [0, 1]$; for $x < min$ and $x > min$, the membership function assumes the value of zero (null). The greater the value of $\mu(x)$, the greater is the truth about the membership of x.

The IP-FULL-FUZZY scheduler employs the uncertainties of application demands by considering edge and vertex weights (I_i and $B_{i,j}$, respectively) of D as triangular fuzzy numbers. The ith task requires $\tilde{I}_i$ instructions with a projected uncertainty level of $\sigma\%$ of this value. Using the notation of fuzzy numbers, the quantity of instructions required by the ith task is represented by $[\,\underline{I_i}, I_i, \overline{I_i}\,]$, where $\underline{I_i}, = I_i \times (1 - \frac{\sigma}{100})$ and $\overline{I_i}, = I_i \times (1 + \frac{\sigma}{100})$. Communication demands are treated in a similar manner, being represented by $\widetilde{B_{i,j}} = [\,\underline{B_{i,j}}, B_{i,j}, \overline{B_{i,j}}\,]$, with $\rho\%$ of projected uncertainty level, i.e., $\underline{B_{i,j}}, = B_{i,j} \times (1 - \frac{\rho}{100})$ and

$\overline{B_{i,j}} = B_{i,j} \times (1 + \frac{\rho}{100})$. The lower the value of σ, the closer to the true computational demand are the weight values furnished by the DAG. Similarly, the lower the value of ρ, the closer to the true communication demand are the weight values furnished in the DAG. Importantly, the values of σ and ρ are not random, but are rather set by the users to express their level of confidence in application demands.

Similarly, the IP-FULL-FUZZY scheduler employs the uncertainties of resource availability by also considering edge and vertex weights (TI_i and $TB_{i,j}$, respectively) of H as triangular fuzzy numbers. The processing capacity of the kth resource is given by $\widetilde{TI}_k = [\underline{TI_k}, TI_k, \overline{TI_k}]$, where $\underline{TI_k}, = TI_k \times (1 - \frac{\chi}{100})$ and $\overline{TI_k}, = TI_k \times (1 + \frac{\chi}{100})$, with $\chi\%$ representing the projected uncertainty level. Similarly, the available bandwidth between resources k and l is given by $\widetilde{TB}_{k,l} = [\underline{TB_{k,l}}, TB_{k,l}, \overline{TB_{k,l}}]$, where $\underline{TB_{k,l}}, = TB_{k,l} \times (1 - \frac{\omega}{100})$ e $\overline{TB_{k,l}}, = TB_{k,l} \times (1 + \frac{\omega}{100})$, with $\omega\%$ projected uncertainty level. The lower the value of χ, the closer is the estimation of the processing availability to the true value in the resources. Similarly, the lower the values of ω, the closer to the true available bandwidth are the estimated ones. The values of TI_k are estimated considering the existing processing load on the resource k, while $TB_{k,l}$ is a fuzzy triangular number which describes the estimations provided by popular available bandwidth estimators such as *pathload* [82] and the *abget* [6] on the network path among k and l.

Concerning the time set $\mathcal{T}$, the following notation is used: $\mathcal{T} = \{1, \ldots, T'_{max}\}$, where $T'_{max} = T_{max}(1 + \frac{\sigma}{100})(1 + \frac{\chi}{100})$ and T_{max} (equation 3.1) is the time that the application would take to execute serially all the tasks on the fastest resource on the target system. The formulation of the IP-FULL-FUZZY scheduler also employs T'_{min} value, where $T'_{min} = T_{min}(1 - \frac{\sigma}{100})(1 - \frac{\chi}{100})$ and T_{min} is the minimal makespan achievable for the DAG. The T_{min} value is obtained when all tasks on the shortest path (SP) of D are executed on the fastest resource, e.g., the time taken to execute all the tasks in the longest dependence chain on the fastest computer. The T_{min} value is defined as follows:

$$T_{min} = \min_{\forall k \in V_H} \left\{ TI_k \right\} \times \sum_{\forall i \in SP} I_i \tag{3.2}$$

T_{max} and T_{min} values can be directly derived by the DAG furnished as input; there is no need to run the application serially on the fastest resources. This is also true for T'_{min} and T'_{max}. In other words, T_{max} and T_{min} are computed as if there were no actual uncertainty level in either the DAG or the target system of the environment execution, and the variables σ and χ introduce the projected uncertainty level in T'_{min} and T'_{max}.

Similar to the IPDT scheduler, the IP-FULL-FUZZY scheduler also solves an integer linear program which seeks the value of the variables $x_{i,t,k} \in \{0, 1\}$, where $x_{i,t,k}$ is a binary variable that assumes a value of 1 if the ith task finishes at time t on the resource k; otherwise this variable assumes a value of 0. The formulation of the IP-FULL-FUZZY scheduler also employs $f_i \in \mathbb{N}^*$, where f_i is an integer variable that only records the time at which the execution of the ith task is finished, e.g.:

$$f_i = \sum_{t \in \mathcal{T}} \sum_{k \in V_H} t \times x_{i,t,k}, \quad \forall i \in V_D \tag{3.3}$$

The IP-FULL-FUZZY is given by a fuzzy optimisation problem that employs fuzzy variables and constraints. This fuzzy formulation is not be present in this document, but for more details, please, see [16]. The fuzzy constraints and the objective function are transformed into corresponding crisp expressions, leading to an equivalent integer programming formulation as follows:

Minimise λ
Subject to:

$$\sum_{t\in\mathcal{T}}\sum_{k\in V_H} x_{j,t,k} = 1 \qquad \text{for } j \in V_D; \qquad (D1)$$

$$x_{j,t,k} = 0 \qquad \text{for } j \in V_D, \quad k \in V_H \quad t \in \{1,\ldots,\left\lceil \underline{I_j}\underline{TI_k} \right\rceil\}; \qquad (D2)$$

$$\sum_{k\in\delta(l)} \sum_{s=1}^{\lceil t-\overline{I_j}\,\overline{TI_l}-\overline{B_{i,j}}\,\overline{TB_{k,l}}\rceil} x_{i,s,k} \;\geq\; \sum_{s=1}^{t} x_{j,s,l} \quad \text{for } j \in V_D, \quad ij \in A_D, \quad l \in V_H, \quad t \in \mathcal{T}; \qquad (D3)$$

$$\sum_{j\in V_D} \sum_{s=t}^{\lceil t+\underline{I_j}\underline{TI_k}-1\rceil} x_{j,s,k} \;\leq\; \tau(k) \qquad \text{for } k \in V_H, \quad t \in \mathcal{T}, \quad t \leq \left\lceil T'_{max} - \underline{I_j}\underline{TI_k} \right\rceil; \qquad (D4)$$

$$x_{j,t,k} \in \{0,1\} \qquad \text{for } j \in V_D, \quad l \in V_H \quad t \in \mathcal{T} \qquad (D5)$$

$$1 - \frac{f_n - T'_{min}}{T'_{max} - T'_{min}} \geq \lambda \qquad (D6)$$

The objective function of the integer programming version of the IP-FULL-FUZZY scheduler maximises the degree of satisfaction $\lambda \in [0,1]$, which is inversely proportional to the makespan of the application (f_n) given by a schedule. Constraint (D6) establishes the relationship between λ and f_n. This objective function jointly with constraint (D6) are equivalent to the objective function of the fuzzy optimisation formulation. The constraints $D1$–$D5$ are equivalent to those constraints $C1$–$C5$. Since applications are described as a set of dependent tasks with a single ending task, the tasks can be ordered so that the nth task is the last one; the makespan is thus given by the ending time of the last task, f_n. Note that, when all parameters σ, ρ, χ, and ω are null, the IP-FULL-FUZZY scheduler is equivalent to the IPDT scheduler.

While the IP-FULL-FUZZY scheduler was not designed to produce schedules in hybrid clouds, it was the only one found in the literature that copes uncertainties of both application demands and resource availability. Therefore, in order to assess the effectiveness of

our proposed procedure, we conduct an experiment that employs the proposed procedure into the scheduler IPDT with the aim of comparing to the IPDT-FULL-FUZZY scheduler, the robust version of the IPDT under uncertainties of both application demands and resource availability. However, as the focus of this experiment is on the uncertainties regarding the available bandwidth in the communication channel connecting clouds, the values of σ, ρ and χ employed by IPDT-FULL-FUZZY were considered null (equal to zero) in all experiments. Only the ω value was varied in this experiment, since only ω reflects the projected uncertainty level of available bandwidth in the scheduling algorithm. The evaluation methodology of this experiment is present in the next section, while the results are showed in Section 3.8.4.

3.8.3 Evaluation Methodology

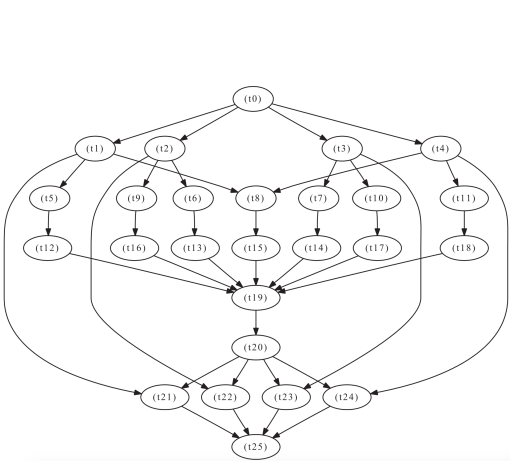
Both the IPDT and the IP-FULL-FUZZY solve an integer linear programming problem and aim to minimise the makespan of the workflow application. While the IP-FULL-FUZZY scheduler employs fuzzy optimisation to cope uncertainties of both application demands and resource availability, its deterministic counterpart (IPDT) does not. Actually, the IPDT is a deterministic scheduler and consequently does not consider any type of uncertainty. In an attempt to improve the effectiveness of the IPDT scheduler in also be able to produce robust schedules under uncertainties of available bandwidth, we employed the proposed procedure into it. In this evaluation, the parameters of σ, ρ and ω used by IPDT-FULL-FUZZY are considered null in this experiment, since these parameters do not reflect the projected level of uncertainty in available bandwidth to produce schedules. Therefore, only the parameter ω was considered to be non-null in this experiment.

For comparative purposes, this experiment employs an evaluation methodology similar to the methodology in which both schedulers were evaluated in [19], and therefore differs from the methodology used Section 3.6. In fact, given the NP-Completeness of the workflow scheduling problem [112], there is no known algorithm that finds the optimal solution in polynomial time deterministically, and schedulers based on integer linear programming (ILP) are tend to usually be time-consuming to produce schedule solutions for large applications [71, 72] (workflow more than 50 tasks, for instance), process that could take from minutes to hours when optimal (or even sub-optimal) schedules are required. Therefore, since both the IPDT and IP-FULL-FUZZY are ILP-based scheduler, this evaluation was then carried out by using the same workflows as in [19] instead of using large workflows presented in Section 3.6.1. Using smaller workflows presented below, the schedules were able to be produced quickly, in a matter of seconds or even minutes.

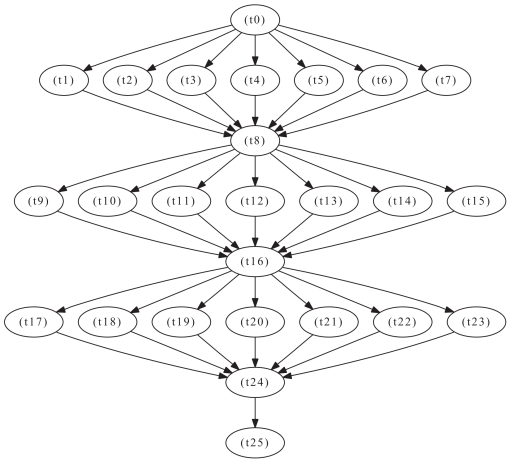
Workflow Applications

In order to use the same DAGs of the assessment of the schedulers IPDT and IPDT-FULL-FUZZY, the DAGs presented here were obtained from [19]. Three DAGs of real applications were employed. The DAG weights were randomly chosen from a uniform distribution with an edge weight mean of 500×10^6 bits, and a vertex (node) weight mean of 2.625×10^{12} instructions. The first DAG was taken from an astronomy application called Montage (the same application as in Section 3.6.1), which is widely used in grid [19, 32,

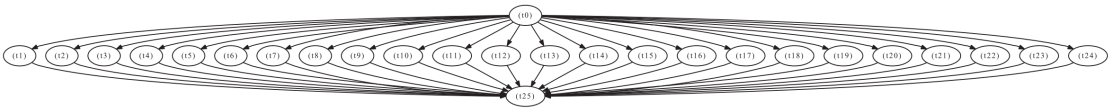
33, 120, 129, 138] and clouds [51, 69–71, 85, 86, 88] environments. Figure 3.15a illustrates the Montage[15] DAG without the weights used in [19]. This DAG contains 26 tasks, with 39 dependencies among them.



(a) Montage DAG



(b) WIEN2k DAG



(c) Modified-WIEN2k DAG

Figure 3.15: DAGs employed in [19]

The second DAG corresponds to an application in quantum chemistry called WIEN2k[16] (Figure 3.15b) developed at the Vienna University of Technology [138]. This DAG represents 26 tasks with 43 dependencies; weights are assigned in the same way as those for the Montage DAG. The third DAG is a modified version of the WIEN2k with 48 dependencies involving parallel tasks in addition to the input and output tasks. This type of 3-level fork-join DAG is very common in applications for image processing. In these applications, the original image is sliced (forked) into n sub-images (pieces) by the input task, and each of n mid-level task executes an image process algorithm in a subarea of the original image given as input data. Lastly, all n processed sub-images are joined by the output task to construct the final image. Because of the *fork* and *join* procedure, this DAG is often called fork-join DAG [35]. From now on, for ease of discussion, we call this modified version of the WIEN2k application as *modified-WIEN2k*.

Execution Environment

As both IPDT and IPDT-FULL-FUZZY schedulers were designed to minimise the makespan of the application, the use of a hybrid cloud of Table 3.2 impaired the assessment of the effectiveness of the proposed procedure in reducing the negative impact of using inaccurate bandwidth estimates introduced by monitoring/measurement tools on the production of

[15] http://montage.ipac.caltech.edu/docs/grid.html
[16] http://susi.theochem.tuwien.ac.at/

schedules. In a hybrid cloud, the available bandwidth on internal links (which connect computers within the cloud) is often larger than the available bandwidth on external links (which connects computers between clouds). This is a reasonable assumption in real environments.

When using a hybrid cloud scenario, such as the scenario presented in Table 3.2 for example, applications were normally scheduled in only one cloud (private or public) regardless the application, scheduler or uncertainties of available bandwidth. Incidentally, depending on the size of the private cloud considered, the application was fully scheduled in this cloud; otherwise, the entire application was usually scheduled in the most powerful (and expensive) public cloud, since this contains faster virtual machines. Because links between clouds tend to be "slower" than links within the cloud, both IPDT and IPDT-FULL-FUZZY schedulers avoided producing schedules that performed data transmissions in slow links since it implied in larger makespans. The experiments were conducted considering a private cloud, similar to the environment employed in [16, 19] We use a private cloud composed of 10 dual-core hosts and all of them connected to each other. We used four different mean weight of the links: 60 Mbps, 80 Mbps, 100 Mbps, and 120 Mbps. The mean weight of the hosts is $\frac{9}{5{,}25\times10^{12}}$ min/instructions (9726 MIPS, which is equivalent to the capacity of an Intel Pentium IV processor). All these values were based on the experiments carried out in [19].

Experimental Setup

The experiment was also carried out in two steps. In the first step, the proposed procedure is applied only in the IPDT scheduler to create its respective multiple linear regression (MLR) equation $f(x, y) = a_1x + a_2y + a_3$ to calculate the $\mathcal{U} = f(x, y)$ value. In the second step, the effectiveness of the IPDT scheduler using the $\mathcal{U}$ value is compared to the IP-FULL-FUZZY scheduler using the ω parameter. The proposed procedure deflates the estimated value of the available bandwidth used as input by the IPDT scheduler. The IPDT-FULL-FUZZY scheduler, however, uses the bandwidth value without the deflation applied by $\mathcal{U}$. Both steps are detailed below and the experimental results are discussed in Section 3.8.4.

First step: This first step is similar to the one described in 3.6.5, which is divided in two substeps. The first one is described in Algorithm 8, where a database that contains historical information about the workflow execution is produced for each type of workflow, available bandwidth, and fixed $\mathcal{U}$ value. In the second sub-step, specifically in Algorithm 9, the values of the coefficients a_1, a_2 and a_3 for the equation $f(x, y) = a_1x + a_2y + a_3$ are produced by employing the multiple linear regression technique using the historical data as input.

Algorithm 8 is employed only using the IPDT scheduler and is similar to Algorithm 7. On line 16, the current DAG $\mathcal{G}_i$, where $\mathcal{G} = \{Montage, WIEN2k, Modified - Wien2k\}$ and $i \in [1; 500]$, is scheduled by the IPDT scheduler using an available bandwidth of $b \in \{60, 80, 100, 120\}$ Mbps deflated by a fixed $\mathcal{U} \in \{0, 5, 15, 25, 35, 45, 50\}$, i.e., $b = b\,(1 - \frac{u}{100})$ in line 15. In lines 17-22, the produced schedule is simulated by the simulator

Algorithm 8 First step of the experimental evaluation – Database creation

1: $A_{pps} = \{Montage, WIEN2k, Modified\text{-}Wien2k\}$
2: $S_{chedulers} = \{IPDT\}$
3: $D_{eflating\_factors} = \{0, 5, 15, 25, 35, 45, 50\}$ ▷ Fixed deflating values in percentagem
4: $B_{andwidths} = \{60, 80, 100, 120\}$ ▷ In Mbps
5: $V_{ariability} = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99\}$ ▷ In percentagem
6: **for each** $\mathcal{S}$ in $S_{chedulers}$ **do**
7: **for each** *app* in A_{pps} **do**
8: Create the database $\mathcal{H}_{\mathcal{G},\mathcal{S}}$
9: $\mathcal{G}$ = DAG that represents the application *app*
10: Create 500 DAGs of $\mathcal{G}$ differing only in the weights of nodes and edges
11: **for** $i = 1$ **to** 500 **do**
12: Calculate T_{max}
13: **for each** b in $B_{andwidths}$ **do**
14: **for each** $\mathcal{U}$ in $D_{eflating\_factors}$ **do**
15: Deflate b using $\mathcal{U}$: $b' = b \times (1 - \frac{u}{100})$
16: Generate a schedule for $\mathcal{G}_i$ using the scheduler $\mathcal{S}$ and a deflated available bandwidth estimate of b' Mbps
17: **for each** p in $V_{ariability}$ **do**
18: Simulate the execution of the workflow using the schedule given in step 16 and assuming b with a variability of up to p: $[b - p\%; b + p\%]$
19: Obtain the makespan m of the workflow given by the simulation in step 18
20: Calculate the speedup as $\tau = \frac{T_{max}}{m}$
21: $\mathcal{H}_{\mathcal{G},\mathcal{S}} = \mathcal{H}_{\mathcal{G},\mathcal{S}} \cup \langle b, \mathcal{U}, p, \tau \rangle$ ▷ Store the 4-tuple in the database
22: **end for**
23: **end for**
24: **end for**
25: **end for**
26: Store the database $\mathcal{H}_{\mathcal{G},\mathcal{S}}$
27: **end for**
28: **end for**

which assumes (simulates or make it happen) that the available bandwidth of b Mbps has a variability of $p = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99\}$, i.e., every time the simulator needs to use the value of the available bandwidth, it takes a new value that is uniformly distributed in the interval $[b - p\%; b + p\%]$. The makespan of the workflow is extracted (line 19) from the simulation step and employed in the calculation of the speedup (line 20), where *speedup* is the ratio between the time taken to process sequentially the tasks of an application (T_{max} value calculated in line 12) and the time to process the same set of tasks in a non-sequential fashion (makespan given by the simulator in line 18). The T_{max} value can be directly derived by the DAG furnished as input; there is no need to run (simulate) the application serially on the fastest resources. Finally, the 4-tuple $\langle b, \mathcal{U}, p, \tau \rangle$ is stored in the database $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ (line 21), which is used as input to the multiple linear regression in the next algorithm (Algorithm 9).

Algorithm 8 generates 500 DAGs for each of the 3 applications in A_{pps} (line 10). The production of DAGs is carried out by using different values in the weight of vertices and edges. Each DAG was simulated using 4 different available bandwidth values, 7 deflating $\mathcal{U}$ values, and 11 p values. Therefore, in this first step, $3 \times 500 \times 4 \times 7 = 42,000$ schedules and $42,000 \times 11 = 462,000$ simulations (or 4-tuples in the database) were carried out in total.

In the second sub-step, Algoritm 9 receives as input a database of executions of the

Algorithm 9 Computation of the coefficients a_1, a_2, and a_3 for the equation $f(x, y)$

1: $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ = Database of previous executions of DAG $\mathcal{G}$
2: $\mathcal{S}$ = the workflow scheduler used to create the database $\mathcal{H}_{\mathcal{G},\mathcal{S}}$
3: $\mathcal{H}'_{\mathcal{G},\mathcal{S}} = \emptyset$
4: **for each** (b', p') pair existing in $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ **do**
5: $ST = None$ ▷ ST = Selected tuple
6: **for all** 4-tuples $\langle b, \mathcal{U}, p, \tau \rangle \in \mathcal{H}_{\mathcal{G},\mathcal{S}}$ as CT **do** ▷ CT = Current tuple
7: **if** $CT.b == b'$ **AND** $CT.p == p'$ **then**
8: **if** $ST == None$ **then**
9: $ST = CT$
10: **end if**
11: **if** $CT.\tau > ST.\tau$ **then**
12: $ST = CT$
13: **end if**
14: **end if**
15: **end for**
16: $\mathcal{U}'$ = Take the $\mathcal{U}$ value from the selected 4-tuple represented by ST
17: $\mathcal{H}'_{\mathcal{G},\mathcal{S}} = \mathcal{H}'_{\mathcal{G},\mathcal{S}} \cup \langle b', p', \mathcal{U}' \rangle$
18: **end for**
19: Call a multiple linear regression by using the $\mathcal{H}'_{\mathcal{G},\mathcal{S}}$ as an input data set and set the function $f(x, y) = a_1 x + a_2 y + a_3$ as a formula to fit the data
20: **Store** the coefficient values a_1, a_2, and a_3

application represented by the DAG $\mathcal{G}$ ($\mathcal{H}_{\mathcal{G},\mathcal{S}}$, line 1), and produces as output the coefficient values a_1, a_2, and a_3 of the linear equation $f(x, y)$ that represents the procedure of $\mathcal{G}$. Such executions are results of the schedules produced by the scheduler $\mathcal{S}$, which in this case is the IPDT scheduler. The equation $f(x, y)$ of the procedure is employed in the second step of the experiment. To produce this equation, the best historical tuples are selected from $\mathcal{H}_{\mathcal{G},\mathcal{S}}$ (lines 4-18). This selection is performed as follows: For each pair (b', p') in the database, the tuple containing the highest *speedup* is selected (lines 11-13) and its $\mathcal{U}$ value is taken (line 16) to compose the 3-tuple $\langle b', p', \mathcal{U}' \rangle$ that is added in the subset $\mathcal{H}'_{\mathcal{G},\mathcal{S}}$ (line 17). This selection is performed because, given b and p values, we want to select which fixed value $\mathcal{U}$ implies the IPDT scheduler in producing schedules that resulted in largest values for *speedup*. After selecting the best tuples, the linear regression technique is called using the subset $\mathcal{H}'_{\mathcal{G},\mathcal{S}}$ as input (line 19), and the coefficient values a_1, a_2, and a_3 of the equation $f(x, y) = a_1 x + a_2 y + a_3$ are produced (line 20). The performance comparison between the IPDT using the procedure $f(x, y)$ and the IP-FULL-FUZZY scheduler is carried out in the next step, described below.

Second step: Algorithm 10 performs the second and last step of this experiment, where the effectiveness of the IPDT scheduler using the proposed procedure is compared to the IP-FULL-FUZZY scheduler. Algorithm 10 employs the same sets A_{pps}, $V_{ariability}$, and $B_{andwidths}$ of Algorithm 8, and also generates 500 new DAGs for each of the 3 applications in A_{pps} (line 7). The production of DAGs is also carried out by using different values in the weight of vertices and edges. Each DAG was simulated using 2 different schedulers, 4 different available bandwidth values, and 11 p values. Therefore, in this second step, $3 \times 500 \times 4 \times 2 \times 11 = 132,000$ schedules and simulations were carried out in total.

The production of schedules is performed as follows. If the scheduler is the IPDT,

the procedure referring to DAG $\mathcal{G}$, $f(x, y)$, is used to deflates the available bandwidth b (lines 14 - 15). By checking the resource monitor tool and making $x = p$ the currently predicted uncertainty and $y = b$, the currently available bandwidth, $\mathcal{U} = f(x, y)$ is calculated and then b is deflated as b'. In other words, the deflation of b is based on a historical of previous events. according to the expected uncertainty p. The deflated available bandwidth b' is furnished as input to the IPDT scheduler to produce a schedule for $\mathcal{G}_i$ (line 16). Once the schedule is produced, it is furnished to the simulator which simulates the execution of the workflow (line 17). The simulation is performed to compute the speedup produced by the IPDT scheduler using the proposed procedure when the currently available bandwidth b had a variation of up to $p\%$ (line 19).

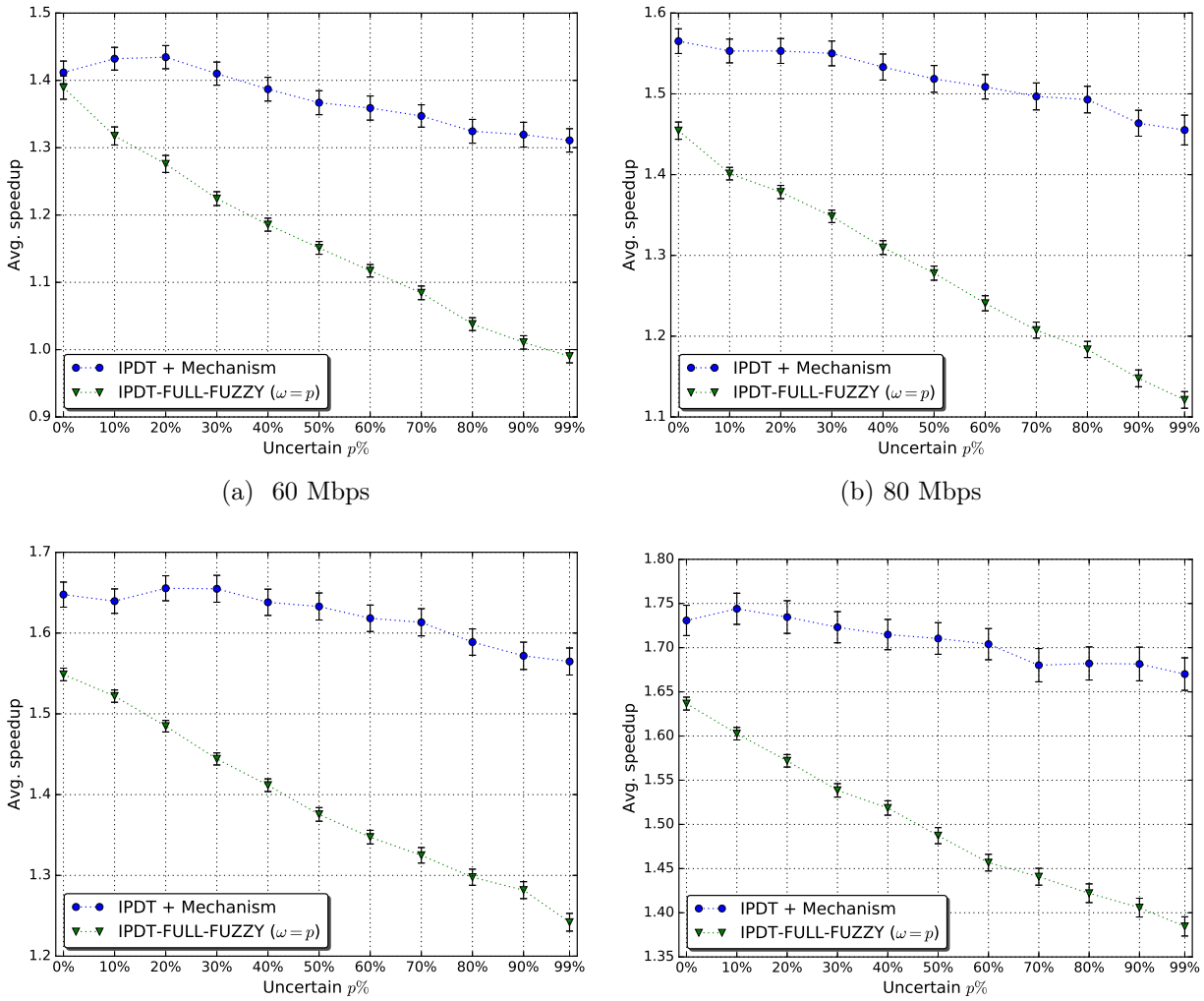
On the other hand, if the scheduler is the IP-FULL-FUZZY, the ω parameter assumes the p value since ω represents the projected level of uncertainty in available bandwidth that the scheduler was designed to meet (line 24). The parameters of σ, ρ and ω used by IPDT-FULL-FUZZY are considered null, since such parameters do not reflect the projected level of uncertainty in available bandwidth. After the schedule is produced in line 25, the simulation of the workflow execution in line 26 is performed in a similar manner from the previous step, and the speedup is then calculated in line 28. In brief, the IP-FULL-FUZZY schedules the application assuming a projected uncertainty level of $\omega\%$ of the available bandwidth furnished by the monitoring/measurement tools.

3.8.4 Results

The IPDT is a deterministic scheduler and consequently does not consider any type of uncertainty to produce schedules. This section presents the results of the simulations carried out to assess the effectiveness of the proposed procedure in providing robustness to the IPDT scheduler in the face of uncertainties in available bandwidth furnished at the scheduling time. The performance of the IPDT scheduler using the proposed procedure is compared to that of the fuzzy scheduler (IP-FULL-FUZZY), which was originally designed to cope those imprecise estimations of available bandwidth. While the IP-FULL-FUZZY "modifies" the IPDT formulation using fuzzy numbers to improve its robustness under such uncertainties, we attempt to enhance the IPDT's robustness using our proposed procedure which uses the original IPDT scheduler without no modification in its formulation. Hereinafter, the term *IPDT-Mechanism* refers to the IPDT scheduler using our proposed procedure. The proposed procedure, the integer linear programming (ILPs) of both IPDT and IP-FULL-FUZZY schedulers, and the simulator were written in the Java programming language. Both ILPs were solved using IBM ILOG CPLEX[17].

Figures 3.16, 3.17, and 3.18 show the average speedup of the schedules derived by the IPDT-Mechanism and IP-FULL-FUZZY for, respectively, the Montage, WIEN2k, and Modified-WIEN2k applications. The speedups are plotted according to the different values of uncertainty p occurred during the simulation step. Each graph contains two curves describing the behaviour of the IPDT-Mechanism and IP-FULL-FUZZY, respectively. As the focus of this study is to compare the effectiveness of both schedulers under uncertainties of available bandwidth when producing schedules for workflow applications,

[17]https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/



(a) 60 Mbps

(b) 80 Mbps

(c) 100 Mbps

(d) 120 Mbps

Figure 3.16: Speedup produced by schedulers for the Montage application as a function of different actual uncertainty levels p in the measured available bandwidth of 60 Mbps, 80 Mbps, 100 Mbps, and 120 Mbps in which p varies from 0% to 99%. ($\sigma = 0\%$, $\chi = 0\%$, and $\rho = 0\%$ for IP-FULL-FUZZY – Montage DAG with 26 nodes and 39 edges)

the speedup values provided by only the IPDT are not evaluated since the IPDT was not originally developed to address the use of imprecise bandwidth information in the production of schedules. Therefore, the aim of this experimental evaluation is to compare which approach (the fuzzy optimisation or the proposed meachanism) "furnish" better robustness to the original IPDT scheduler.

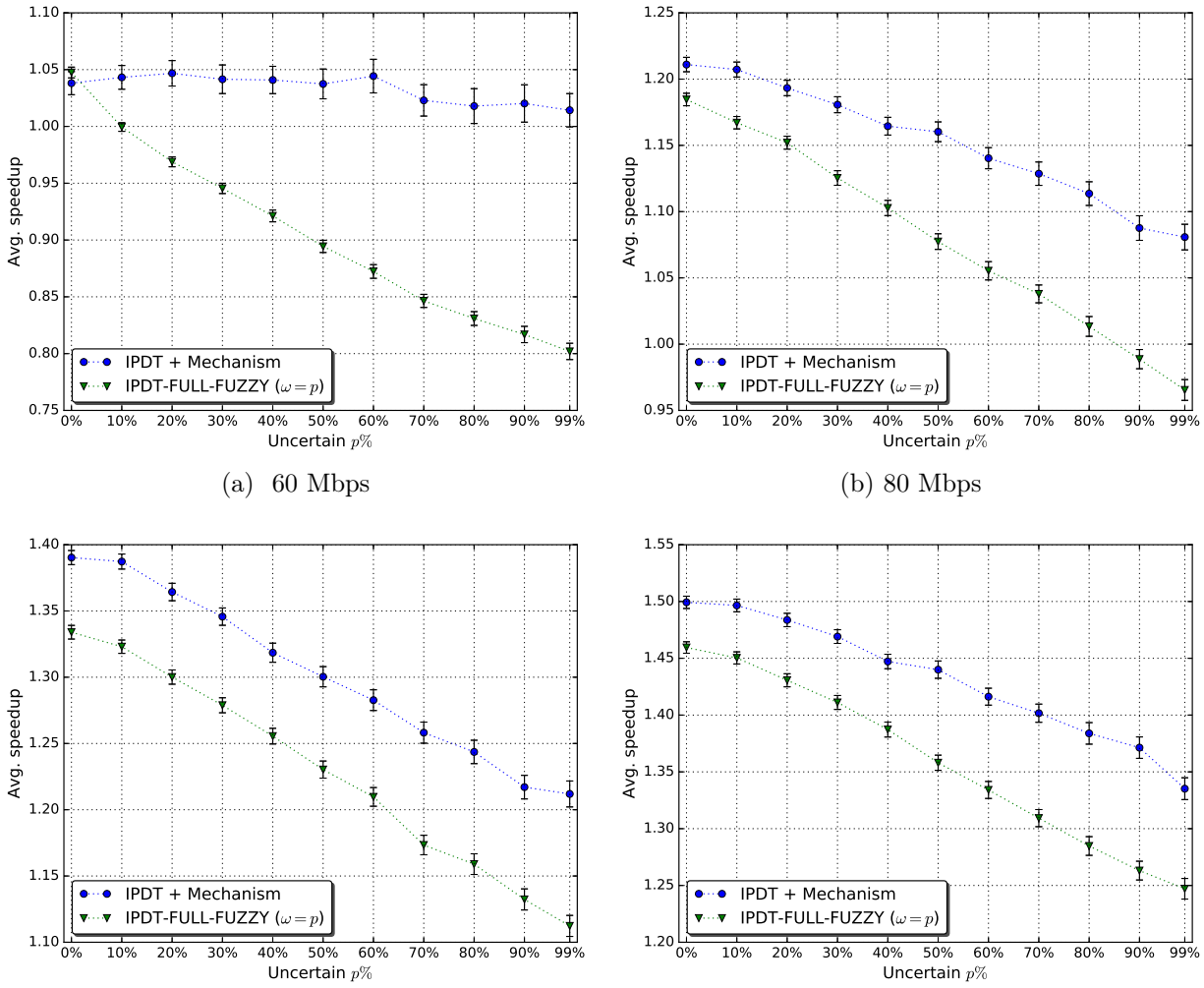
Figures 3.16a–3.16d show, respectively, the average speedup for the Montage application when the measured available bandwidth of 60 Mbps, 80 Mbps, 100 Mbps, and 120 Mbps, were employed in the experiment. As can be seen in these figures, the performance of both schedulers fed by inaccurate information underwent some degradation as the degree of uncertainty p increases. In fact, this degradation was an expected behaviour for both robust scheduling approaches as the degree of uncertainty p increases. However, one approach is more robust than the other if it can slow down this degradation as the uncertainty increases. As Figures 3.16a–3.16d, the decay rate of the average speedup provided

by the IP-FULL-FUZZY is remarkable sharper than does that of the IPDT-Mechanism, showing that the IPDT-Mechanism achieved a better performance than does the IP-FULL-FUZZY. For instance, when the measured available bandwidth was 100 Mbps, the average speedup had a gradual decline of 5% from $p = 0\%$ to $p = 99\%$ when the IPDT-Mechanism is used, and a sharp decline of approximately 20% when the IP-FULL-FUZZY is employed.

Regardless the degree of uncertainty p and the measured available bandwidth in the evaluation considering the Montage application depicted in Figure 3.16, the IPDT-Mechanism scheduler produces higher speedup values than those produced by the IP-FULL-FUZZY. Figure 3.16 also reinforces that the uncertainty-supported IP-FULL-FUZZY scheduler undergoes more degradation in speedup than that produced by the IPDT-Mechanism. Even in scenarios with high uncertainties, a remarkable significant difference between both robust approaches was found for all measured values of the available bandwidth. For example, for a measured available bandwidth of 120 Mbps and $p = 99\%$, the IPDT-Mechanism reveals an average speedup of approximately 22% higher than those furnished by the IP-FULL-FUZZY. Figure 3.16 thus reinforces the fact that the effectiveness of the proposed procedure relies on its ability to further improve the robustness of the IPDT scheduler, even in high levels of uncertainty, than does the IP-FULL-FUZZY that employs the fuzzy optimisation into the IPDT for this purposes.

Figures 3.17a–3.17d depict, respectively, the average speedup for the WIEN2k application when the measured available bandwidth of 60 Mbps, 80 Mbps, 100 Mbps, and 120 Mbps, were employed in the experiment. Figures 3.17a–3.17d show the average speedup of the evaluation results as a function of the degree of uncertainty p projected in the available bandwidth estimates during the simulation step. The results shown in Figure 3.17 reinforces those in Figure 3.16 in which the IPDT-Mechanism produces higher speedup values than those produced by the IP-FULL-FUZZY for all cases. For instance, for an uncertainty degree of 50% and a measured available bandwidth of 80 Mbps, the average speedup produced by the IPDT-Mechanism was approximately 10% higher than that produced by the IP-FULL-FUZZY. Furthermore, a remarkable difference between both schedulers is also observed in scenarios with high degree of uncertainties. For instance, when $p = 99\%$, the results achieved by the IPDT-Mechanism were 14% higher than the results produced by the IP-FULL-FUZZY. The trend observed in Figure 3.16 can also be observed in Figure 3.17, where the uncertainty-supported IP-FULL-FUZZY scheduler undergoes significant degradation in its performance than that of the IPDT-Mechanism as the degree of uncertainty of available bandwidth estimates increases.

Moreover, when the measured available bandwidth was of 60 Mbps (Figure 3.17a), the rate of decay of the speedup produced by the IP-FULL-FUZZY is much sharper than that arising when the IPDT-Mechanism is employed. In this particular case, the average speedup produced by the IP-FULL-FUZZY had a sharper decline of approximately 30% when p ranged from 0% to 99%, while a modest decline of approximately only 5% had been produced by the IPDT-Mechanism. Actually, for this particular case, it might also be noted in Figure 3.17a that when the measured available bandwidth was of 60 Mbps, the IP-FULL-FUZZY produced speedup values smaller than one (≤ 1) for most of the cases, showing a significant degradation in its performance in the presence of uncertainty p. On



(a) 60 Mbps

(b) 80 Mbps

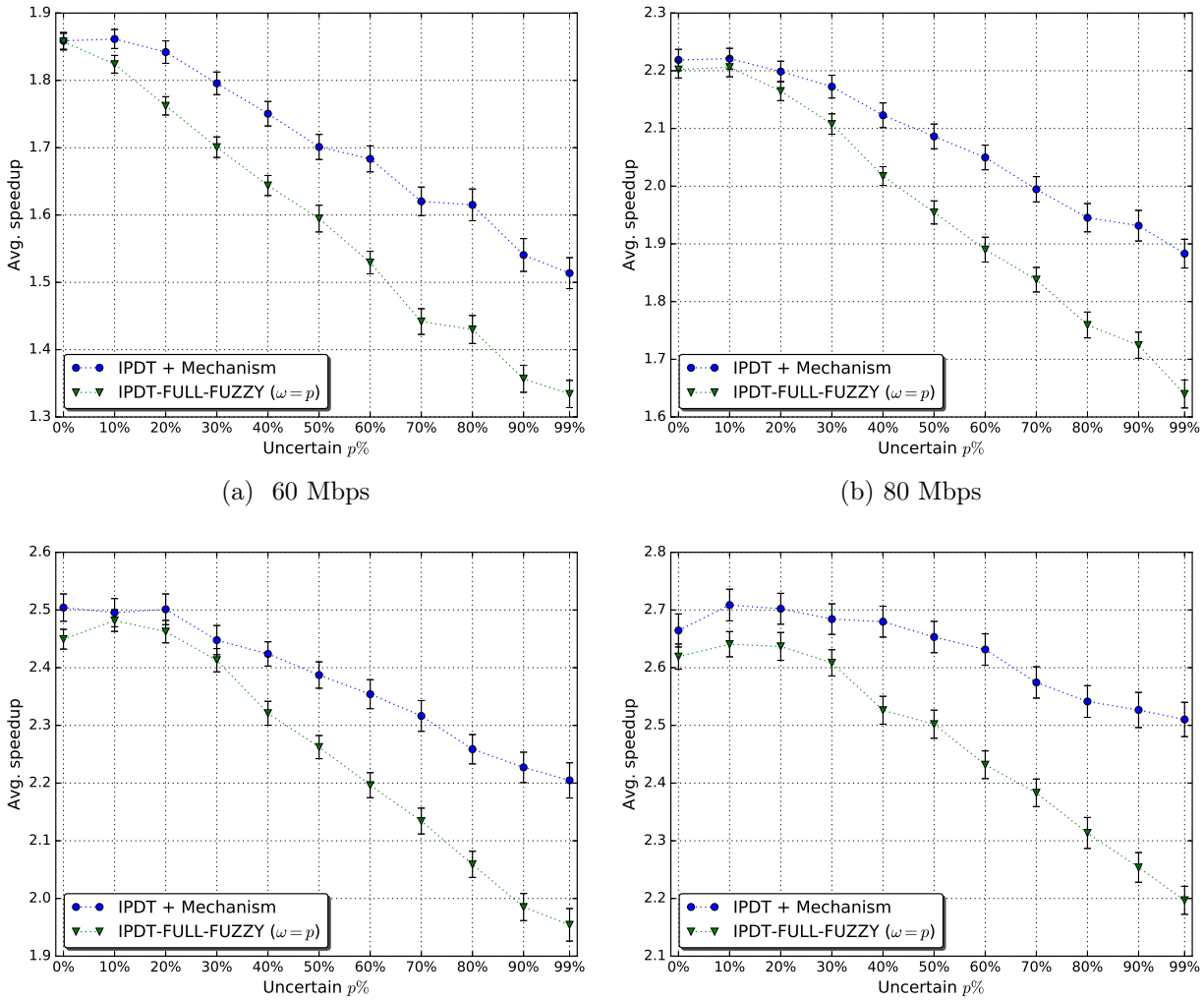
(c) 100 Mbps

(d) 120 Mbps

Figure 3.17: Speedup produced by schedulers for the WIEN2k application as a function of different actual uncertainty levels p in the measured available bandwidth of 60 Mbps, 80 Mbps, 100 Mbps, and 120 Mbps, in which p varies from 0% to 99%. ($\sigma = 0\%$, $\chi = 0\%$, and $\rho = 0\%$ for IP-FULL-FUZZY – DAG with 26 nodes and 43 edges)

the other hand, the IPDT-Mechanism was able to produce schedules in which the speedup values were modestly greater than one (≥ 1) regardless the degree of uncertainty p. Seemingly, the WIEN2k application generates huge amounts of data during its execution and a measured available bandwidth of only 60 Mbps was not quite enough for both schedulers in finding schedules that result high speedup values. For instance, the speedup values in Figures 3.17b-3.17d is higher than those in Figure 3.17a, showing that the execution of the WIEN2k application requires a scenario where the available bandwidth is greater than 60 Mbps to achieve higher speedup values. Due to this, any variability on the bandwidth availability of 60 Mbps was sufficient to keep speedup values around to one – such as those achieved by the IPDT-Mechanism – or even below to one – such as those given by the IP-FULL-FUZZY.

Incidentally, since the speedup value is the ratio between the time taken to process sequentially the tasks of an application and the time to process the same set of tasks in a



(a) 60 Mbps

(b) 80 Mbps

(c) 100 Mbps

(d) 120 Mbps

Figure 3.18: Speedup produced by schedulers for the modified-WIEN2k application as a function of different actual uncertainty levels p in the measured available bandwidth of 60 Mbps, 80 Mbps, 100 Mbps, and 120 Mbps in which p varies from 0% to 99%. ($\sigma = 0\%$, $\chi = 0\%$, and $\rho = 0\%$ for IP-FULL-FUZZY – DAG with 26 nodes and 48 edges)

non-sequential fashion, the schedules produced by the IP-FULL-FUZZY when $p > 0$ are not quite useful since it is better to run all the tasks sequentially in only one resource. On the other hand, the IPDT-Mechanism was able to produce better schedules in all scenarios with any degree of uncertainty since most of the speedup values were greater than 1, evidencing the ability of the IPDT-Mechanism in amortising the presence of uncertainty p in the available bandwidth estimates to produce schedules. Especially in scenarios when the bandwidth availability was not favourable to yield high speedup values, as shown Figure 3.17 when the measured available bandwidth was of 60 Mbps.

Figures 3.18a–3.18d depict the averages speedup values of the next experiment when the WIEN2k application had been evaluated with a measured available bandwidth of 60 Mbps, 80 Mbps, 100 Mbps, and 120 Mbps, respectively. Similar to previous results, uncertainties of available bandwidth also result in speedup degradation in both robust scheduling approaches. As shown in Figure 3.18, the decay rate of the average speedup

provided by the IP-FULL-FUZZY is also sharper than does that of the IPDT-Mechanism, showing that the IPDT-Mechanism could be able to achieve better performance than does the IP-FULL-FUZZY in the evaluation of the modified-WIEN2k application. In other words, in this last experiment, the effectiveness of the IPDT-Mechanism in reducing the negative impact of using uncertain bandwidth estimates to achieve schedules that results in high speedup values than does that of the uncertainty-supported IP-FULL-FUZZY. For instance, for a high uncertainty degree of 99% and a measured available bandwidth of 120 Mbps, the average speedup produced by the IPDT-Mechanism was approximately 16% higher than that produced by the IP-FULL-FUZZY. Again, the trend observed in Figure 3.18 mirrors Figures 3.16 and 3.17, evidencing that the IP-FULL-FUZZY undergoes more degradation in achieving high speedup values than the IPDT-Mechanism scheduler in the presence of any degree of uncertainty p in the available bandwidth estimates.

Lastly, the average speedup values provided by the IPDT-Mechanism decay much faster for the WIEN2k (Figure 3.17) and adapted-WIEN2k (Figure 3.18) applications than those produced for the Montage application (Figure 3.16). This result is somehow dependent of the DAG topology since the average speedup values of DAGs with high degree of parallelism (WIEN2k and adapted-WIEN2k) undergoes more degradation than those with high degree of pipelines (Montage) as the degree of uncertainty increases.

3.8.5 Discussion

The difference in performance between both schedulers lies in the fact that, to produce robust schedules in the presence of imprecise bandwidth estimates, the IPDT-Mechanism (the IPDT scheduler using the proposed procedure) uses a deflated value of the measured available bandwidth according to an expected uncertainty of up to p% of such measurement, while the IP-FULL-FUZZY scheduler uses the measured available bandwidth as a triangular fuzzy number with a projected uncertain level of $\omega = p\%$. Although both schedulers tend to produce robust schedules under uncertainties of bandwidth availability, the results presented in this section have indicated that the effectiveness of the IPDT-Mechanism relies on its ability to cope with a high level of uncertainty than does the uncertainty-supported IP-FULL-FUZZY [16, 19]. Note that this section only compares the robustness of both the IPDT-Mechanism and IP-FULL-FUZZY schedulers under uncertainties of available bandwidth, which is the purpose of this thesis. Other types of comparison between them are left as future work.

According to the experimental evaluation carried out in this study, the performance of the IP-FULL-FUZZY fed by imprecise bandwidth information underwent significant speedup degradation than did the one of the IPDT-Mechanism. In general, the results show that the speedup degradation produced by the IPDT-Mechanism was approximately 25% lower than those occurred in the schedules produced by the IP-FULL-FUZZY, evidencing the ability of the proposed procedure in enhancing the robustness of deterministic schedulers in the face of uncertainties of the available bandwidth estimates. In this case, the determinism scheduler was the IPDT scheduler [16, 19].

It is worth mentioning that both the IPDT-Mechanism and IP-FULL-FUZZY attempt to enhance the robustness of the IPDT scheduler under uncertainty of available band-

width, since IPDT is a deterministic scheduler and it was not designed to address such uncertainties of bandwidth availability. Therefore, we argue that the performance of the IPDT using our procedure to cope the presence of any degree of uncertainty in bandwidth estimates was better than that of the uncertainty-supported IP-FULL-FUZZY, which was originally designed to improve the robustness of the IPDT under uncertainties of both application demands and resource availability. Hence, the trend observed in Section 3.7 can also be observed in Section 3.8.4, where the proposed procedure achieved its purpose in enhancing the robustness of deterministic schedulers by allowing them in starting making robust scheduling decisions in the face of uncertainties stemming from using unreliable monitoring/measurement networking tools.

Although the IPDT-Mechanism has shown to be more effective than the IP-FULL-FUZZY, we cannot depreciate the usability of the IP-FULL-FUZZY scheduler. Incidentally, our proposed procedure relies on a historical information of previous workflow executions to calculate a proper $\mathcal{U}$ value that is used to deflate the measured available bandwidth. In the absence of such historical information, the IPDT-Mechanism can not be applied, which leaves the IP-FULL-FUZZY a good alternative to produce robust schedules under uncertain bandwidth availability. The IP-FULL-FUZZY is based on fuzzy optimisation for the consideration of the uncertainties of available bandwidth, and because of this, it does not rely on a historical information since the uncertainties are modelled as fuzzy triangular numbers which can assume different uncertainty levels. Moreover, the IP-FULL-FUZZY also copes other types of uncertainties, such as uncertainties of application processing demands and resource processing availability (in the matter of processing power of resources). As this study aims to minimise the negative impact of using uncertain bandwidth estimates to produce schedules in a scenario composed of hybrid clouds, the uncertainties of application processing demands and resource processing availability were not considered when the IP-FULL-FUZZY was applied in the evaluation.

3.9 Final Remarks

Resources in hybrid clouds, consisting of private and public resources, are normally connected through Internet links whose available bandwidth fluctuates considerably because of the shared communication infrastructure [16, 34, 121, 122, 132]. The use of imprecise information about the available bandwidth may cause a scheduler to generate schedules that result in slow application execution time (makespan) and/or poor application performance [34, 67, 68, 102]. Actually, one of the main issues faced in scheduling problems is the quality of the information that is provided as input data to the scheduler to produce a schedule that does not degrade the application execution performance [14, 16, 44, 85].

Uncertainties of available bandwidth can be a result of imprecise measurement and monitoring network tools and/or their incapacity of estimating the real value of the available bandwidth during the application scheduling time [16]. In general, bandwidth estimations are given in ranges rather than deterministic values [14, 82, 91, 92, 108, 116, 121–123, 137], and as a consequence, schedules produced by deterministic schedulers may make decisions that lead to ineffective executions of applications. The results described

in this chapter show that applications can have their execution performance significantly degraded as the degree of variability on the bandwidth availability increases, especially when the applications are modelled as workflows are expected to be executed in a hybrid cloud scenario.

Indeed, most deterministic schedulers in the literature neglect the presence of variability in the bandwidth estimates in their design. The study present in this chapter aims to provide a procedure that can be used by these schedulers in order to enhance their robustness under uncertainties of input values, especially bandwidth availability. We argue that, by using imprecise information, deterministic schedulers may make decisions (produce schedules) that lead to ineffective executions of applications. We also argue that the negative impact of using imprecise values may be reduced thought the adoption of procedures that take into consideration the uncertainties of such input values.

This chapter presented a proactive approach (procedure) that refines the bandwidth estimates given by monitoring tools [14] *before* furnish it to the scheduler. The proposed procedure produces a deflating multiplier value that is applied to the measured available bandwidth, which is given by such tools, before being furnished as an input to the scheduler. According to an expected uncertainty p of the available bandwidth measurement, the procedure deflates the measured bandwidth to prevent misleading information from affecting the performance of the scheduler in producing poor schedules. The results presented in this chapter evidence the production of robust schedules using ordinary deterministic schedulers when our procedure was employed. The results have indicated the performance of these schedulers fed by imprecise information underwent significant degradation than the performance of the same schedulers when our procedure was used. It is worth mentioning that the proposed procedure does *not* modify the original algorithm of the schedulers, it is just a proactive procedure that assists them in producing robust schedules under imprecise bandwidth estimates.

Experiments through simulations carried out with several workflows and deterministic schedulers have evidenced the effectiveness of the procedure in improving the robustness of these schedulers in the presence of a high degree of uncertainty in the available bandwidth. For example, Section 3.7 described the experiments carried out using the HCOC and adapted-PSO-based schedulers with our proposed procedure. Both schedulers are classified as a deadline- and cost-aware scheduler, and therefore they focus in minimising the monetary execution costs as well as in satisfying specified deadlines for such executions. According to the simulations, the employment of the proposed procedure on both schedulers was able to reduce the number of disqualified solutions (solutions that missed the deadline) by up to 70% with costs reduced by up to 10%. In fact, the results evidence that the number of qualified solutions when the procedure was considered was significantly higher than that achieved when the procedure was not considered, besides to making executions even cheaper. The proposed procedure has been able to considerably improve the effectiveness of both schedulers under imprecise bandwidth estimates, since they have started producing robust schedules that underwent remarkable less degradation in performance than that of given by the same schedulers without using our procedure.

Section 3.8 presented the experiments carried out on the IPDT scheduler. We called IPDT-Mechanism when the IPDT used our proposed procedure. The IPDT is a deter-

ministic scheduler that maximises the speedup of schedules, where speedup is the ratio between the time taken to process sequentially the tasks of an application and the time to process the same set of tasks in a non-sequential fashion. The results given by the IPDT-Mechanism was compared and contrasted with the IP-FULL-FUZZY scheduler, a robust version of the IPDT under uncertainties of both application demands and resource availability. The IP-FULL-FUZZY scheduler was designed based on the IPDT and it employs fuzzy optimisation for the consideration of the uncertainties of the input values. In general, the performance of the IP-FULL- FUZZY fed by imprecise bandwidth estimates underwent significant speedup degradation than did the one of the IPDT using the procedure (IPDT-Mechanism). The schedules produced by the IPDT-Mechanism were compared with those returned by IP-FULL-FUZZY. According to the results, the speedup produced by the IPDT-Mechanism is on average 25% higher than that of the IP-FULL-FUZZY, evidencing the effectiveness of the proposed procedure in enhancing the robustness of the IPDT is better than that of the IP-FULL-FUZZY that employs fuzzy optimisation to address uncertainties in the input data.

The performance of workflow execution in hybrid clouds depends largely on the available bandwidth in inter-cloud links since this bandwidth determines the data transfer time between tasks residing in different clouds. As workflow applications may generate huge amounts of data during their executions, the procedure proposed in this chapter seems to be attractive for hybrid cloud environments, where the available bandwidth in inter-cloud links may vary and it is difficult to be estimated precisely in advance.

Algorithm 10 Computation of the coefficients a_1, a_2, and a_3 for the equation $f(x, y)$

1: $A_{pps} = \{Montage, WIEN2k, Modified\text{-}Wien2k\}$
2: $S_{chedulers} = \{IPDT, IP\text{-}FULL\text{-}FUZZY\}$
3: $B_{andwidths} = \{60, 80, 100, 120\}$ ▷ In Mbps
4: $V_{ariability} = \{0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 99\}$ ▷ In percentagem
5: **for each** *app* in A_{pps} **do**
6: $\mathcal{G}$ = DAG that represents the application *app*
7: Create new 500 DAGs of $\mathcal{G}$ differing only in the weights of nodes and edges
8: **for** $i = 1$ **to** 500 **do**
9: **for each** b in $B_{andwidths}$ **do**
10: **for each** p in $V_{ariability}$ **do**
11: **for each** $\mathcal{S}$ in $S_{chedulers}$ **do**
12: **if** $\mathcal{S} == IPDT$ **then**
13: Calculate T_{max}
14: Get the $f(x, y)$ equation (procedure) referring $\mathcal{G}$ and calculate the deflating $\mathcal{U} = f(b, p)$ value
15: Deflate b using $\mathcal{U}$: $b' = b \times (1 - \frac{u}{100})$
16: Generate a schedule for $\mathcal{G}_i$ using the IPDT scheduler and a deflated available bandwidth estimate of b' Mbps
17: Simulate the execution of the workflow given by the schedule in step 16 assuming b with a variability of up to p: $[b - p\%; b + p\%]$
18: Obtain the makespan m of the workflow given by the simulation in step 17
19: Calculate the speedup as $\tau = \frac{T_{max}}{m}$
20: **else if** $\mathcal{S} == IP\text{-}FULL\text{-}FUZZY$ **then**
21: Calculate T'_{min}
22: Calculate T'_{max}
23: $\rho = \chi = \sigma = 0$ ▷ Projected uncertainies is null (0%) for these parameters
24: $\omega = p$ ▷ Projected uncertainty for the available bandwidth estimate is equal to the expected (that it will be simulated in the simulation step) uncertainty
25: Generate a schedule for $\mathcal{G}_i$ using the IP-FULL-FUZZY scheduler and an available bandwidth estimate of b Mbps with ω% representing the projected uncertainty level of b
26: Simulate the execution of the workflow given by the schedule in step 25 assuming b with a variability of p: $[b - p\%; b + p\%]$
27: Obtain the makespan m of the workflow given by the simulation in step 26
28: Calculate the speedup as $\tau = \frac{T'_{max}}{m}$
29: **end if**
30: **end for**
31: **end for**
32: **end for**
33: **end for**
34: **end for**

Chapter 4

A Flexible Scheduler for Workflow Ensembles

The ever-growing data and compute requirements of large-scale applications have led to extensive research on how to efficiently schedule and deploy them in distributed environments. With the emergence of the latest distributed systems paradigm, cloud computing brings with it tremendous opportunities to run scientific applications modelled as workflows at low costs without the need of owning a specialised and dedicated infrastructure [56]. It provides a virtually infinite pool of resources that can be acquired, configured, and used as needed and are charged on a pay-per-use basis [147]. While scientific workflows have become the mainstream to conduct large-scale scientific research, cloud computing has emerged as an alternative cost-effective computing paradigm for researchers. However, along with these benefits come numerous challenges that need to be addressed to efficiently execute these large-scale scientific applications.

Today's computational, experimental, and observational sciences rely on computations that involve many related tasks. The success of a scientific mission often hinges on the computer automation of these workflows [129]. Actually, workflows are frequently used to model large-scale distributed scientific applications. They facilitate the expression of multi-step task workloads in a manner which is easy to understand, debug, and maintain [96]. In fact, large-scale scientific applications are normally modelled as several workflows grouped into *ensembles* [84, 97, 129]. Workflows in an ensemble typically have similar structure, distinguishing only by their sizes (number of tasks), input data, and individual computational task sizes. There are many applications that require scientific workflow ensembles. For example, Montage is an application for astronomy research [119] that creates a science-grade astronomical image-mosaics using data collected from several telescopes. The Galactic Plane project [119], in turn, uses several Montage workflow applications into ensembles to combine several mosaics of different region of the sky to generate a single large image-mosaics of the entire sky. The Galactic Plane ensemble consists of 17 workflows, each of which contains 900 sub-workflows [97]. Another ensemble example is the Periodograms application [136], which searches for extrasolar planets by detecting periodic dips in the light intensity of their host star.

Workflows in an ensemble may differ not only in their parameters, but also in their priority. CyberShake application [43, 97], for example, uses ensembles to generate seismic

hazard maps. Each workflow in a CyberShake ensemble generates a hazard curve for a particular geographic location, and several hazard curves are combined to create a hazard map. However, some geographic locations may be in heavily populated areas or have sensitive facilities such as power plants, while others may be less important. Due to this, researchers typically prioritise the workflows of an ensemble so that critical and important workflows are finished first. This enables them to see critical results early, and helps them to choose the most important workflows to run in situations when the time and financial resources available for computing are strictly limited. Once the execution priorities for workflows in an ensemble are given, for instance, it is the scheduler's role to find a suitable schedule that satisfies the stipulated constraints and achieve determined goals for the workflow ensemble execution. Minimise the workflow ensemble execution time (overall makespan[1]) under budget constraint, for example, is one type of scheduling problem that a scheduler may have to face.

Indeed, the most common objective of a workflow scheduling algorithm (scheduler) is to minimise the schedule length (makespan) [33]. The schedule length is directly responsible for the execution time of the workflow ensemble. However, when workflows in an ensemble share computing resources, besides the minimisation of the overall makespan, there might be conflicts among each workflow's execution that must be observed and tackled to guarantee the efficiency of the entire ensemble execution. To fairly distribute the workflows for execution in a way that no one will be allocated to the resources with a disadvantage when compared to each other in terms of execution time is a job for the scheduler. When scheduling several workflows simultaneously on the same set of resources, each workflow may experience a *slowdown* in its execution, when compared to the runtime of the same workflow when it has the resources on its own. Due to resource sharing, it may be interesting to distribute the workflows on the resources in a way that the processing capacities can be fairly shared among them. In this case, the scheduler should consider fairness to share the resources equally among the workflows of an ensemble. Recently, with the emergence of cloud computing, scientists regards cloud computing as a potentially attractive source of low-cost computing resources [102]. By provisioning resources on demand, clouds give more flexibility in creating a controlled and managed computing environment, which facilitates the running of applications since scientists can lease resources at anytime and anywhere on a pay-per-use basis [107]. However, the use of a large number of cloud resources may result in small ensemble execution time at the expense of a high monetary cost. In this case, cost-efficient schedules may be required to save expenses, and to achieve this goal, it is advisable to use a cost-aware scheduler.

A scheduler is therefore a module responsible to make the distribution of application's workloads[2] on resources. There are numerous ways to accomplish such distribution, but the scheduler is normally designed to optimise certain specific objectives to make an efficient distribution (schedule). For each objective, a different scheduler is selected/designed/created to achieve the desired goal. For example, to share the resources equally among the workflows of an ensemble, the use of a fairness-aware scheduler should be

[1]The sum of the makespans of all workflows.

[2]In this case, to execute an application as a workflow ensemble, each workflow of the ensemble can be seen as a workload of the application execution.

more appropriate to schedule the tasks on resources [33, 149]. For other purposes, such as saving cost expenses with executions of applications using pay-as-you-go resources offered by cloud providers, the use of cost-aware schedulers would be more suitable in this case [69, 104]. On the other hand, when it is necessary to speed up applications' executions, the use of a makespan-aware scheduler is more convenient since it aims to accomplish schedules that minimise the overall makespan (execution time) of the application [32, 130]. Or when there are deadline constraints (desired application's execution time), a deadline-aware scheduler should be more appropriate in this case [97]. For each scenario, there is a more suitable scheduler to achieve the expected objectives and meet the expected constraints to run applications. It is the user's job to find/ create the appropriate scheduler to schedule his/her applications according to his/her needs.

Minimisation of makespan, maximisation of fairness, and minimisation of cost are common examples of objective functions for a workflow scheduler in the literature. Schedulers are generally designed to optimise one or two objective functions; however, they are *not* designed to allow the replacement of these objective functions, specially the schedulers that cope with workflow ensembles. Normally, for each objective established for the scheduling procedure of an application, a different scheduler is often used to address such goal. The flexibility of replacing the objective function is not common to occur in current schedulers because these functions are usually tightly attached to the scheduling algorithm. To address this issue, this chapter focuses on developing a scheduler for workflow ensembles that promotes this flexibility, facilitating the user's job in finding/creating a different scheduler for his/her needs.

This chapter introduces therefore a novel flexible scheduler for workflow ensembles (and multiple workflows[3]). A scheduler is considered *flexible* when it allows the replacement of the objective function according to the user's needs. Similar to a modular design, the objective function can be seen as a "module" that can be switched when necessary, making the scheduler usable in various scenarios, such as saving cost expenses with executions of ensembles using cloud providers or sharing the resources equally among the workflows of an ensemble. Therefore, the contribution of this chapter is to propose a flexible scheduler to schedule scientific applications modelled as workflow ensembles that allows the replacement of its objective function according to the user's needs. The proposed scheduler employs a method called Particle Swarm Optimisation (PSO), a population-based heuristic search algorithm [90]. By having a population (called a swarm) of candidate schedule solutions (called particles), PSO basically uses a fitness function to select the best solution within the swam. Thus, by having different fitness functions, PSO can select the best solution in different ways, providing the desired flexibility. To the best of our knowledge, this is no work in the literature that proposes a flexible scheduler that addresses scientific applications as a set of workflows. Despite many work has been done in workflow scheduling, the scheduling of workflow ensemble stills being an open problem, and only initial studies exist in the literature [33, 41, 60, 97, 110, 149].

The flexibility of the proposed PSO-based scheduler is then accomplished by modelling

[3]Section 4.1 clarifies the differences and similarities between the terms *workflow ensembles* and *multiple workflows*. Both terms, within this thesis, refer to applications composed of set of workflows and are used interchangeably.

the workflow scheduling problem as a PSO and developing a fitness function that acts as a scheduler's objective function. To highlight the proposed flexibility, four different fitness functions (scheduler's objective function) are employed on the PSO (scheduler). The fitness function f_1 aims to drive PSO in achieving solution that minimises the overall makespan of the ensemble, while the functions g_1 and g_2 aim to maximise the fairness among the workflows of the ensemble. Lastly, the function h_1 attempts to drive PSO in producing low-cost schedules. As shown further ahead in Section 4.6, and reinforced in Section 4.7, the desired flexibility for the proposed scheduler has been achieved since each fitness function could produce schedules that satisfied its corresponding objective, which makes the proposed scheduler usable in various cloud scenarios.

This chapter is organised as follows. Section 4.1 clarifies the difference between the terms *workflow ensemble* and *multiple workflows*, since the proposed scheduler is developed in order to address the scheduling a set of workflows which includes both types of workflow set. Related work is describe is Section 4.2, while the assumptions of the scheduling problem addressed in this chapter are discussed in Section 4.3. The proposed PSO-based flexible workflow scheduler is introduced in Section 4.4 and evaluated in Section 4.5, while the results are discussed in Section 4.6. As the proposed flexible scheduler does not have any mechanism (or procedure) in its PSO-based design to address uncertain input information, especially the information about the available bandwidth, Section 4.7 uses the procedure presented in Chapter 3 to provide a certain level of robustness for the schedules produced by the proposed flexible scheduler under uncertainties of available bandwidth. Section 4.7 also includes a performance comparison of the flexible scheduler when the procedure is and is not used, and shows the effectiveness of the procedure in providing a certain level of robustness for the flexible scheduler in the presence of imprecise bandwidth measurements. Section 4.8 provides final remarks and concludes this chapter.

4.1 Workflow Ensemble vs. Multiple Workflows

There is a small difference between the terms workflow ensemble and multiple workflows (Figure 4.1). The term *workflow ensemble* usually represents an entire scientific application that comprises a set of inter-related workflows [84, 97, 129]. Workflows in an ensemble typically have a similar structure, distinguished only by their sizes (number of tasks), input data, and individual computational task sizes. These inter-related workflows are grouped together because their combined execution produces a desired output for an application. Due to this, we define the term *multiple workflows* to express a several different types of non-interrelated workflow applications that are merely grouped into a set. The output of each workflow of a set of multiple workflows is not be correlated with other outputs, i.e., the outputs are not necessarily combined to compose a unique desired output like it occurs in workflows in an ensemble. The term *multiple workflows* can also be defined as a *a set of multiple (non-interrelated) workflow ensembles*. In both cases, workflows within a set are independent to each other, i.e. there are no links (edges) connecting them. Without loss of generality, multiple workflows can been seen as a set of

non-interrelated workflow ensembles of one workflow each. Both terms, within this thesis, refer to applications composed of set of workflows and are used interchangeably.

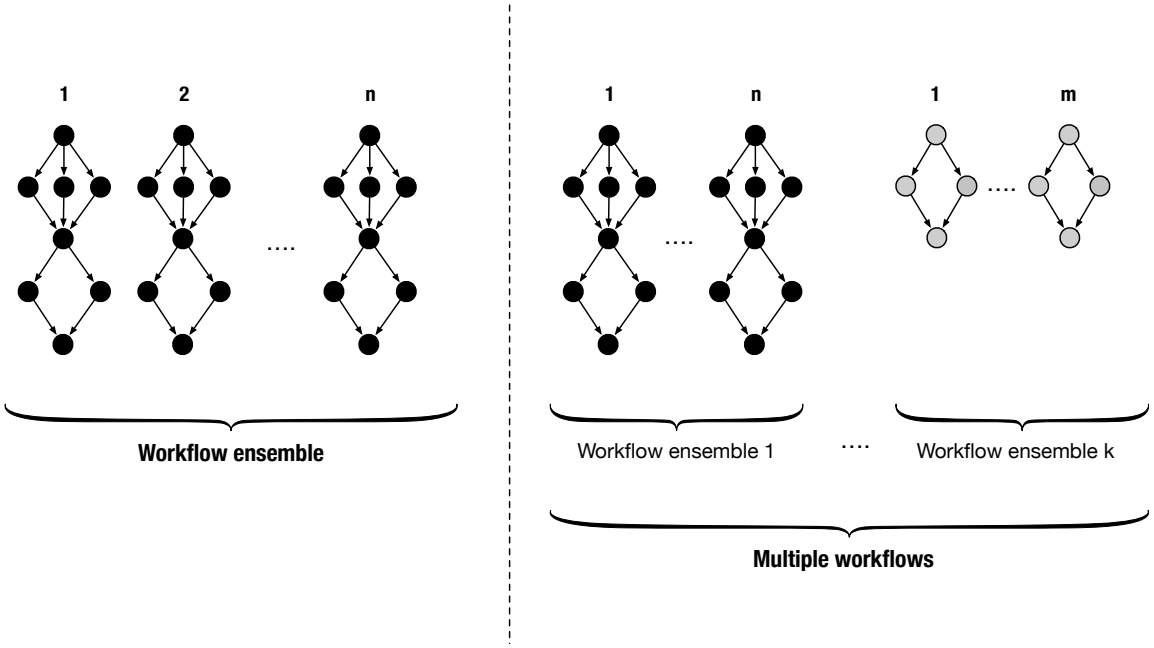


Figure 4.1: Comparison of workflow ensemble (left) and multiple workflows (right).

Examples of Applications modelled as workflow ensemble: Most of the scientific applications require workflow ensembles instead of just using one workflow. Scientists often use ensemble workflows, and research in ensemble management is also needed to support the end-to-end computational methods [56]. The CyberShake application [43, 97], for example, uses ensembles to generate seismic hazard maps. Each workflow in a CyberShake ensemble generates a hazard curve for a particular geographic location, and several hazard curves are combined to create a hazard map. In a study carried out in 2013[4] in California, the CyberShake application was used to generate a set of hazard maps over 286 sites that required an ensemble of 2288 workflows. Another example is the Montage[5], an application for astronomy research [57, 119] that creates science-grade astronomical image-mosaics using data collected from several telescopes. For example, the Galactic Plane project [119] uses several Montage workflow applications grouped into an ensemble to combine several mosaics of different regions of the sky to generate a single large image-mosaic of the entire sky. In fact, the Galactic Plane usually consists of 17 workflows, each of which contains 900 inter-related sub-workflows [97].

Another[6] interesting example of the usability of workflow examples is the Periodogram application [136] for astronomy research. This application processes time-series data collected by NASA's Kepler mission[7]. The Kepler satellite uses high-precision photometry

[4] `http://scec.usc.edu/scecpedia/CyberShake_Study_13.4`

[5] `https://pegasus.isi.edu/application-showcase/montage/`

[6] For more examples of scientific applications modelled as workflow ensembles, such as the LIGO and Epigenomics, please, access the website `https://pegasus.isi.edu/application-showcase/`

[7] `http://kepler.nasa.gov/`

to search for exoplanets transiting their host stars. Similarly to CyberShake and Montage applications, users of the Periodograms often need several inter-related (similar) workflows with different parameters to searches for extrasolar planets by detecting periodic dips in the light intensity of their host star. For example, in a study carried out in 2010[8], the study project used a data set containing 210,664 light curves, which record how the brightness of a star changes over time. According to the researchers, the analysis of light curves to identify the periodic dips in brightness that arise from transiting exoplanets requires the calculation of periodograms, which reveal periodic signals in time-series data and estimate their significance. Generating periodograms is a computationally intensive process that requires high-performance, distributed computing, such as grids or clouds. In order to support the analysis of Kepler's 210,664 light curve dataset, the application consisted of 43 sub-workflows, each of which contains approximately 5,000 tasks. According to the researchers, this workflow ensemble usually required approximately 66 days of sequential computation (one sub-workflow after another), consuming 17 GB of input data, and producing 105 GB of output data in 1.2 million output files.

4.2 Related Work

Workflows have been widely used by scientist communities to model complex scientific applications in order to run them on distributed heterogeneous resources [64,109,129,149], such as in the grid computing [5, 13, 15, 17, 19, 26, 28, 29, 31–33, 44, 62, 64, 106, 120, 130, 131, 138, 141, 144] as well as in the cloud computing [20, 21, 27, 34, 35, 54, 63, 67–72, 75, 78, 91, 94, 97, 98, 107, 110, 117, 118, 127, 135, 140, 148]. The use of workflows facilitates the data processing of such experiments since it allows the fragmentation of large-scale applications into multi-step computational tasks connected in series as a pipeline [129]. Examples of such tasks may include: retrieve data from a database, reformat the data, process the data, run analyses, post-process results, join results, etc.

Large-scale scientific experiments have been adopting workflows to assist complex data analysis that need to process large amounts of data and to run complex simulations and experiments efficiently [41, 63, 111, 118, 127]. However, workflow scheduling is an NP-Complete problem [112] and its optimisation version is NP-Hard. Due to this, most efforts in scheduling algorithms are concentrated on heuristics [31–33, 35, 120] and meta-heuristics [73–75, 104] to obtain feasible schedule solutions. There are also some works which use approximation algorithms [64,109] and linear programming [16,19,69,71,72,107] approaches to achieve (sub) optimal solutions. Indeed, the problem of scheduling a single workflow on heterogeneous resources has been well studied and a number of solutions have been proposed [5, 13, 15, 17, 19–21, 26–29, 31, 32, 34, 35, 44, 54, 62–64, 64, 67–72, 75, 78, 91, 94, 98, 106, 109, 110, 117, 118, 120, 127, 129, 131, 135, 138, 140, 141, 144, 148]. However, with the emergence of large-scale workflow applications nowadays, it is quite common to consider a scenario in which more than one workflow need to be scheduled simultaneously on the resources. The scheduling of a set of workflows simultaneously is still an open issue and only few relevant studies have been developed [33, 41, 60, 97, 110, 149].

[8] `https://pegasus.isi.edu/application-showcase/periodograms/`

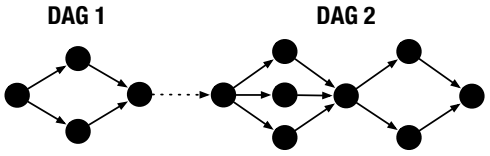


Figure 4.2: An illustration of the naïve solution to schedule multiple DAGs, where each DAG is schedule independently one after another by a ordinary single-DAG scheduler. The dashed line can be seen as a costless edge among DAGs, joining all DAGs into an only one when connecting the exit node of a DAG with the entry node of the next one

Indeed, an obvious and naïve solution of scheduling multiple workflows is put all workflows in a queue according to the order they arrived, and uses any *single*-workflow algorithm scheduling (such as the HEFT [130], PCH [29], and HCOC [35]) to schedule independently one workflow after another (Figure 4.2). Using this approach to schedule a set of workflows, where each workflow starts its execution only after the execution of the previous one has finished. The potential drawback with this primitive approach is that the resources may be idle for some periods of time waiting for the data dependency, which may result in a very long overall-makespan[9]. Moreover, this naïve procedure may result in a waste of money if the resources are leased through pay-per-use basis.

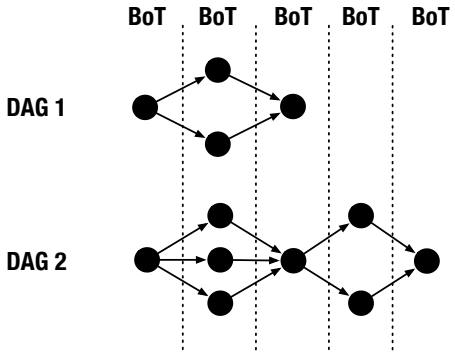


Figure 4.3: An example of selecting independent tasks of multiple workflows to create a bag-of-tasks (BoT), where each task within a BoT has no dependence between them.

Another solution for scheduling multiple workflows is to schedule all ready tasks of all the workflows by using an independent-tasks scheduling algorithm, like the schedulers in [101,148]. Figure 4.3 shows an example of selecting independent tasks of multiple workflows, where each task between dotted lines has no dependence between them. Although this approach is quite simple, it does not deal with the data locality problem [48, 107]. As the scheduler is not aware of the resource's localisation where the predecessor and successor of each task be executed, this approach can result in high network utilisation. If the resources are nearby, in terms of network distance, this approach is very likely to work adequately and could produce good schedules. In fact, the advantage of this approach is that the scheduler does not need to handle the task dependencies, reducing the complexity of developing a scheduling algorithm to cope multiple workflows. However, due to the

[9]The execution time (makespan) of all workflows.

high variability of the available bandwidth in Internet links, this approach may also not be adequate to work in a hybrid cloud.

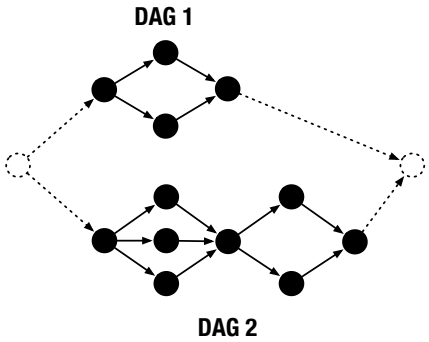


Figure 4.4: An example of merging two DAGs into a single one by adding an entry and an exit node. Dotted lines represent costless nodes and edges.

Scheduling of multiple workflows in community environments is usually performed using one of four approaches. Two of them were the naïve approaches previously described. The third approach is based on a merging of all tasks from multiple workflows into one larger workflow (Figure 4.4), and schedule this large workflow using any single-workflow scheduling algorithm. Finally, the fourth approach uses a prioritisation of workflows (or parts of workflows) in a set to find the most efficient order of scheduling. Our proposed flexible scheduler uses this last approach. However, regardless the approach employed by the scheduler, the idea of elaborating a scheduler that tackles multiple workflows is to take advantage of the idle time left on resources to run tasks of other workflows. If the scheduler is developed to cope multiple workflows, it can make a schedule in a way that, while the resource is waiting for the completion of data transfers of tasks executions of one workflow, the resource can process the tasks of another workflow, minimising, therefore, the resource idleness (or maximising the resource utilisation) and minimising the overall makespan. Figure 4.5 highlights the empty spaces of the schedule of Figure 3.3 that the scheduler can use to schedule tasks of other workflows.

Zhao and Sakellariou proposed in [149] heuristics to schedule multiple DAGs (workflows) onto heterogeneous resources on grids. These heuristics focus not only on minimising the overall makespan but on achieving fairness as well. The term fairness is defined on the basis of the slowdown that each DAG would experience (as a result of sharing the resources with other DAGs as opposed to having the resources on its own). Basically, the heuristics depend on the use of any single-DAG scheduling algorithm, such as the schedulers HEFT [130], PCH [29], and HCOC [35], since several DAGs are merged into a single-DAG. In other words, the authors make the scheduling of multiple DAGs by merging them into a single-DAG, and schedule this resulting DAG using an ordinary single-workflow scheduler. The merging of all the DAGs is carried out by creating one *entry* and one *exit* node and connecting them to all DAGs, as shown Figure 4.4. Four different ways to merge all the DAGs into a single one are proposed by the authors. In contrast to the flexible scheduler proposed in this chapter, these heuristics are not flexible since they do not allow the replacement of the objective function. The authors focus

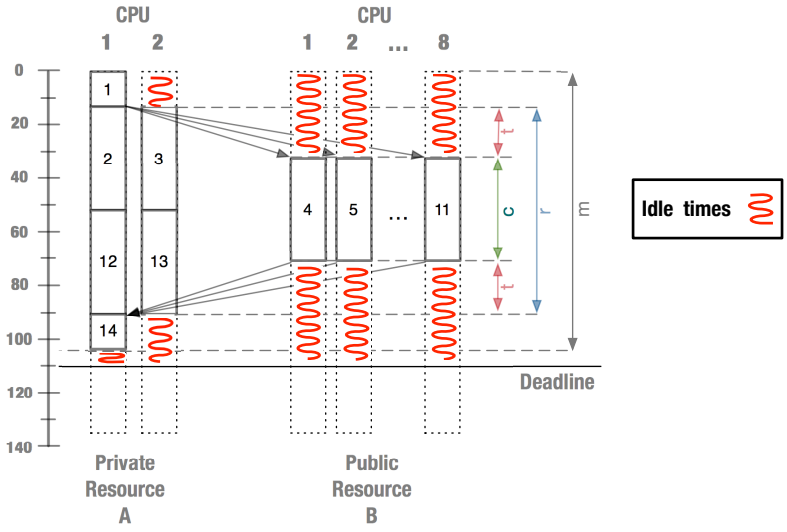


Figure 4.5: Highlighting the empty spaces of the schedule of Fig. 3.3 that can be leveraged to schedule tasks from other workflows. Empty spaces in a schedule result in resource idleness unless it is used to schedule the tasks of another workflow. Empty spaces may represent a waste of money since the resource might be leased through pay-per-use basis.

only in minimising the overall makespan but achieving fairness. However, these heuristics cannot be applied to minimise, for instance, the overall monetary costs for executions of multiple workflows in a hybrid cloud, as this thesis focuses intently.

When scheduling several DAGs simultaneously on the same set of resources, each DAG may experience a slowdown in its execution because of competition for resources. A schedule is assumed to be fair when each DAG of the ensemble experiences equal (or similar) slowdown in its execution. Due to this, Bittencourt and Madeira also described [33] strategies for scheduling multiple DAGs on grids in terms of makespan and fairness. The DAGs are not merged into a single one in this case, but the strategies also depend on the use of a single-DAG scheduling algorithm to produce the schedules, such as the PCH scheduler [29]. Basically, the authors schedule the DAGs independently, one after another similar to the naïve approach. However, during the scheduling of a DAG, they employed a gap search or backfilling technique to schedule tasks on resources. This method takes advantage of the gaps left on resources between scheduled tasks as a result of communication dependencies. In other words, this technique attempts to fill the gaps left in the scheduling of the previous DAG with the tasks of the current DAG (Figure 4.5). They concluded that interleaving the workflows leads to good average makespan and provides fairness when multiple workflows share the same set of resources. The authors experimentally demonstrated that it is possible to achieve fairness based on similar distribution of the slowdown amongst the DAGs, without this happening at the expense of the overall makespan. Nevertheless, our scheduler is flexible to comply with the objective of minimising the overall monetary execution costs in a hybrid cloud, for example. Also, our scheduler provides a feature that allows, for example, the customisation of the scheduler's objective function to achieve fairness schedules satisfying budget constraints; the scheduler in [33] does not allow this flexibility.

Yu and Shi proposed in [145] a dynamic scheduling strategy for multiple workflow applications in a grid environment. They proposed a scheduling algorithm based on a heuristic that ranks the tasks of a workflow in order of execution priority, where the entry node is the node with the highest priority for execution while the exit node is the least priority node. This rank strategy is similar to the rank used by the one-workflow scheduler HEFT [130]. Iteratively and dynamically, the scheduler schedules the highest priority tasks of each DAG without requiring merging the workflow applications a priori, attempting to maximise resource utilisation. While the *Planner* calculates the priority for tasks in a DAG, the *Executor* re-prioritises all tasks ready to execute in a real time fashion and sent for execution. A task is considered ready when its parents have completed execution and have provided the files necessary for the tasks' execution. In particular, the work in [145] schedules workflows in a best effort manner without considering quality of service (QoS) requirements, such as workflow deadline or budget. In contrast, due to the best effort manner, the work in [145] schedules workflows on-the-fly and consequently it can produce poor schedules when it is necessary to maximise the fairness or minimise the monetary execution cost, for instance. Due to the flexibility of our proposed scheduler, we can easily adapt it to satisfy several types of QoS.

Often, the actual execution of applications differs from the original schedule following delays such as those caused by resource contention and other issues in resource performance. These delays have further impact when running multiple workflow applications due to inter-task dependencies. Chen et al. addressed in [49] the problem of scheduling multiple workflow applications on large-scale distributed computing systems (LDCSs), such as grids, by employing a single-workflow heuristic-based scheduler, also proposed by them in [47]. They employed a deadline-based strategy to schedule and reschedule workflows in order to guarantee tasks' deadlines, allowing tasks to be rescheduled to other resources, leaving resource availability to more urgent tasks. Basically, the authors schedule multiple workflows using an adaptive approach that rescheduling (adapts) schedules of single-workflows already produced by a single-workflow scheduler. Using a rescheduling approach, tasks are rearranged individually within certain time slot boundaries (deadline) so that the time constraints of each task (i.e. data time transfer) are kept without needing to totally reconsider the schedules of other dependent-tasks. The authors' evaluation showed that the tasks could be finished before their deadlines and the overall resource utilisation was improved. In contrast, since our proposed scheduler is flexible, we can transform it to be similar to the scheduler in [47] but the other way around is not possible. The scheduler in [47] is inflexible and cannot become a cost- or fairness-aware scheduler easily without major modifications.

Combining workflows into sets before scheduling helps not only to increase effectiveness of resource utilisation, but also contributes to more qualitative schedules. Using additional available information about properties of a scheduling problem can further improve the scheduling result of a static scheduler. Bochenina et al. considered in [36] the problem of scheduling multiple workflows under time restrictions specified by both providers of resources (time windows) and users (soft deadlines, i.e., preferable finishing times). They focused the scheduling problem into community environments system such as grids. These environments are characterised by a static structure of the set of

resources and free-of-charge usage model. They stated that the differences between utility and community environments are reflected in different goals of scheduling process. In utility environments, a scheduling algorithm should guarantee the satisfaction of QoS constraints for a dynamic resource pool; in community environments, an algorithm deals with a static set of resources and it cannot ensure that all users' constraints will be met. Therefore, they considered the scheduling of multiple workflows to partially available heterogeneous resources. The problem is how to fill free time windows with tasks from different workflows, taking into account users' requirements (deadline) of the urgency for schedules. As it is a static scheduling, the inputs of the scheduling algorithm include full information about workflows in the set, execution times of tasks using different resources, communication times between the tasks and for each resource. In contrast, the object function of the scheduler in satisfying the soft deadlines is strictly linked with the scheduling algorithm, making difficult to the scheduler in satisfying other types of objectives as our proposed scheduler is able to do.

Malawski et al. presented in [97] three strategies (heuristics) to schedule a several inter-related workflows grouped into ensembles (workflow ensembles) under budget and deadline constraints on clouds. The goal of these strategies is to maximise the amount of user-prioritised workflows in an ensemble that can be completed on clouds given budget and deadline constraints. To maximise the number of executions of priority workflows, the algorithms decide which workflows in an ensemble are admitted or rejected for execution in order to satisfy such constraints. Pietri et al. extended the work [97] in [110] by including energy constraints along with either budget or deadline constraints. Both works also experimentally demonstrated that it is possible to maximise the number of higher-priority workflows in the ensemble under the given constraints. Bryk et al. complemented the work [97] in [41] by not ignoring file transfers between workflow tasks as both [97] and [110] neglected it. In fact, Malawski et al. and Pietri et al. assume that the file transfer time between tasks is either negligible or included in task runtimes. While this assumption may be correct for some types of workflows, for data-intensive workflows, for example, it may lead to incorrect or overly optimistic schedules as this thesis has been deeply addressing. Due to this, Bryk et al. have observed that file transfers in intra-cloud and inter-cloud links may have a significant impact on ensemble execution; for some data-intensive applications, up to 90% of the execution time is spent on file transfers. To solve this issue, they attempted to minimise the number of transfers by taking advantage of data caching and file locality. In contrast, these schedulers may not produce good schedules to minimise the overall makespan or maximise the fairness since they are inflexible to be re-configured to meet other objectives like these.

Scheduling multiple large-scale parallel workflow applications on heterogeneous computing systems like hybrid clouds is a fundamental NP-complete problem that is critical to meeting various types of quality of service (QoS) requirements. Due to this, Duan et al. address in [60] the scheduling problem of large-scale applications inspired from real-world, characterised by a huge number of homogeneous and concurrent bags-of-tasks. The authors formulated the scheduling problem as a sequential cooperative game and propose a communication and storage-aware multi-objective algorithm that optimises two user objectives (execution time and economic cost) while fulfilling two constraints (network

bandwidth and storage requirements). Basically, the authors applied multi-objective optimisation of time and cost under storage bandwidth and capacity constraints using heuristics based on game theory. To schedule multiple workflows, the authors select independent tasks of multiple workflows to construct a bag-of-task (BoT), where all tasks in BoT can be executed in parallel since they are independent from each other. Similar to previous works, the scheduler in [60] is also not flexible to attend several types of objective functions that the user might need. Also, our scheduler can be customised in a way that it can cope a multi-objective function like the scheduler in [60].

Many scientific applications are represented as workflows of tasks. As a result, workflow scheduling has become an important and popular topic of research. Workflow scheduling algorithms have been widely studied and there are numerous works on algorithms for scheduling single workflows onto generic execution units. This includes algorithms like HEFT [130] and many others. While these algorithms provide good results for single workflows, they are not directly applicable to the execution environment we consider in this research, where an application composed of several (inter-related) workflows is executed in a hybrid cloud where the resources can be provisioned and deprovisioned on demand. With increasing focus on large-scale applications on hybrid clouds, it is important for a workflow management service (scheduler/broker) to efficiently and effectively schedule and dynamically steer execution of applications.

The analysis of related work is summarised in Table 4.1, where we classify the methods based on 7 categories. The *infrastructure* type can be a grid or a cloud, where a grid includes utility grids and heterogeneous computing systems, where cost can be also relevant [40]. *Application* types of interest are workflow ensembles or multiple workflows. However, single-workflow is on the table only to highlight that all of the works in the table can handle single-workflow as well. The *workload* can be static, where the scheduling process can plan the tasks in advance, or dynamic, where the algorithm has to deal with unpredictable stream of workflows arriving. The *provisioning* methods are divided into dynamic and static approaches, where static means that the provisioning resource plan is prepared prior to execution. The *methods* to develop scheduling algorithms for workflow ensembles are basically split into two types: heuristics, including various list scheduling techniques based on graph analysis and meta-heuristics that employed, for instance, particle swarm optimisation or evolutionary algorithms such as genetic algorithms. The related work addresses several types of optimisation *objectives* (minimisation or maximisation) for the scheduling problem, including time (makespan), monetary cost, resource utilisation, or fairness, and there are examples of multi-criteria optimisation among them. In fact, most of the scheduling approaches also consider *constraints* for optimisation, and the most commonly addressed ones are budget, deadline, and slowdown. Other constraints, such as bandwidth, storage capacity or energy utilisation, are also present. Finally, scheduler can be developed to be *robust* under uncertainties of available bandwidth. As Chapter 3 focused, the available bandwidth in inter-cloud links can fluctuate considerably since Internet links are shared resources. This fluctuation contributes to imprecision estimates of the available bandwidth, and furnishing such inaccurate estimates as an input to a scheduler usually results in the production of poor schedules.

As the Table 4.1 shows, our proposed scheduler is designed especially for cloud-based

infrastructures, but it can easily be adapted to work in grid-based infrastructures as well. The scheduler is designed to cope applications modelled as single-workflow, multiple-workflow, and workflow ensembles. We assume a scenario where the scheduler is designed to be called by a broker when it receives an application for execution (Figure 3.2). Once the application is received, the broker calls the scheduler to start the decision-making process of the scheduling the application. Due to this, the workload of our scheduler is considered static since the application (known in advance) is one the input data to the scheduler to produce a schedule. After the schedule is produced, the broker rents resources from the public cloud according to the the scheduler's decision, i.e., according to the information extracted from the schedule, and then proceeds for the execution of the application. Therefore, the resources are statically provisioned prior the execution.

A workflow scheduler is considered *flexible* in this thesis when it allows the replacement of the objective function according to the user's needs. For instance, similar to a modular programming[10], the objective function can be seen as a "module" that can be switched when necessary, making the scheduler usable in various scenarios. Unlike other scheduling algorithms, which are based on heuristics, our proposed scheduler does not need major modifications to accept different types of objective function. Basically, the flexibility of the proposed scheduler is possible because it employs particle swarm optimisation (PSO), a popular meta-heuristic tool widely used due to its simplicity and its effectiveness in solving a wide range of NP-Complete problems at a low computational cost. Basically, the flexibility of the proposed PSO-based scheduler is accomplished by modelling the scheduling problem as a PSO and developing a fitness function (module) that acts as an objective function of the scheduler. By having a population (called a swarm) of candidate schedule solutions (called particles), PSO uses a fitness function to select the best solution within the swam. Therefore, by having different fitness functions, PSO can select the best solution in different ways as well as with different types of constraints. The proposed PSO-bases scheduler is presented in Section 4.4.

[10]Modular programming is a software design technique that emphasises separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality [59].

Table 4.1: Summary table of related work that regards the problem of scheduling multiple workflows.

Work	Grid	Cloud	Single workflow	Multiple workflows	Workflow Ensemble	Static	Dynamic	Static	Dynamic	Heuristic	Meta-heuristic	Minimise overall makespan	Maximise overall cost	Maximise fairness	Maximise # of workflows executions	Minimise # of data file transfer	Maximise resource utilisation	Bandwidth	Budget	Deadline	Energy	Slowdown	Storage	Under uncertainties of available bandwidth
	Infrastructure		**Applications**			**Workload**		**Provisioning**		**Method**		**Objectives**						**Constraints**						**Robust**
Zhao and Sakellariou [149]	+	-	+	+	-	+	-	+	-	+	-	-	-	+	-	-	-	-	-	-	-	+	-	-
Bittencourt and Madeira [33]	+	-	+	+	-	+	-	+	-	+	-	-	-	+	-	-	-	-	-	-	-	+	-	-
Yu and Shi [145]	+	-	+	+	-	-	+	-	+	-	-	-	-	-	-	+	-	-	-	-	-	-	-	-
Chen et al. [49] and [47]	+	-	+	+	-	+	-	-	+	+	-	-	-	-	-	-	+	-	-	+	-	-	-	-
Bochenina et al. [36]	+	-	+	+	-	+	-	+	-	+	-	-	-	-	-	-	+	-	-	+	-	-	-	-
Malawski et al. [97]	-	+	+	-	+	+	-	+	+	+	-	-	-	-	+	-	-	-	+	+	-	-	-	-
Pietri et al. [110]	-	+	+	-	+	+	-	+	+	+	-	-	-	-	+	-	-	-	+	+	+	-	-	-
Bryk et al. [41]	-	+	+	-	+	+	-	+	+	+	-	-	-	-	-	+	-	-	+	+	+	+	-	-
Duan et al. [60]	-	+	+	+	-	+	-	+	-	+	-	+	+	-	-	-	-	+	-	-	-	-	+	-
Genez et al. [73]	-	+	+	+	+	+	-	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+
Genez et al. [74]*	-	+	+	+	+	+	-	+	-	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+*

* Using the procedure presented in Chapter 3

Although the works discussed in this section present various strategies to schedule a set of workflows concurrently, none of them is considered flexible since they do not allow the replacement of the objective function. In contrast to related work, this chapter proposes a flexible scheduler in which the objective of the scheduler can be developed according to the user's needs. For example, our workflow scheduler can be configured to reduce the overall makespan while minimise the fairness, as the fairness-aware schedulers presented in [33, 149], or to produce schedules under budget or deadline constraints, as the cost- and deadline-aware described in [41, 97, 110]. Therefore, our flexible-aware scheduler is designed to handle the scheduling of a set of workflows simultaneously on a set of heterogeneous resources that are provisioned and deprovisioned on demand by cloud providers. Importantly, none of the schedulers in Table 4.1 address the uncertainty of available bandwidth to produce schedules. None of them are robust under uncertainties of available bandwidth. To solve this issue, we applied our procedure, described in Chapter 3, in our proposed PSO-based flexible scheduler in Section 4.7.

4.3 Assumptions of the Proposed Flexible Scheduler

Before describing the proposed flexible scheduler, we need to clarify some important assumptions of the scheduling context. This chapter addresses the scheduling scientific applications on cloud resources by using a PSO-based flexible scheduler. It is assumed that the applications as ensembles of inter-related scientific workflows, which are modelled as direct acyclic graphs (DAGs). Cloud resources are assumed to be virtual machines (VMs) that are provisioned and deprovisioned on-demand and are billed by the hour, with partial hours being rounded up, similar to the resource leasing model from Amazon's Elastic Compute Cloud (EC2)[11]. It is considered that tasks of a workflow may run concurrently on multi-core VMs, and there is no preemption during the execution of the tasks. Finally, as the schedules are produced statically by our proposed PSO-based scheduler, the live migration of executions of tasks among VMs is not supported in this study.

4.4 Flexible Workflow Scheduler

This section presents the proposed flexible scheduler based on particle swarm optimisation (PSO), a meta-heuristic as it makes few or no assumptions about the problem being optimised but it can search very large spaces of candidate solutions at a low computational cost. To introduce the proposed scheduler, this section is organised as follows. A brief overview of the PSO procedure is detailed in Section 4.4.1, while Section 4.4.2 shows how a particle is used to represent a schedule of an ensemble of workflows. In general, scheduling a set of workflows simultaneously (workflow ensembles) involves some scheduling policies, such as by which task of which workflow the scheduler should start the scheduling. Due to this, some scheduling policies employed by the proposed scheduler are detailed in Section 4.4.3. The feature of flexibility is accomplished by the possibility of developing a specific fitness function that addresses the user's needs. To highlight this,

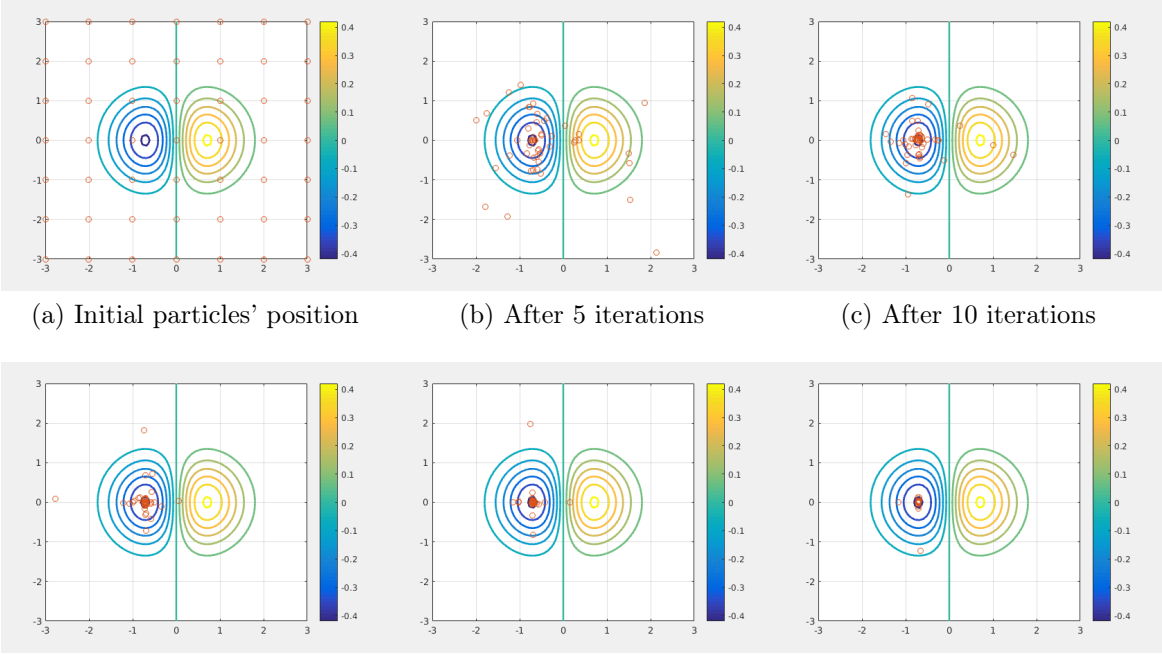
[11] `https://aws.amazon.com/ec2/pricing`

Section 4.4.4 presents four different types of fitness function that acts as the scheduler's objective function. Four different (conflicting) fitness functions were developed here. One function regarding the minimisation of overall makespan, two functions for the maximisation of fairness, and one function for the minimisation of costs. These fitness function are summarised in Section 4.4.5.

4.4.1 Particle Swarm Optimisation

This section highlights the main PSO's features to facilitate understanding to the proposed flexible scheduler, but an overview of the Particle Swarm Optimisation (PSO) is described in Section 3.4.1. PSO is a population-based heuristic global search algorithm introduced by Kennedy and Eberhart [90], inspired by social behaviour of bird flocking or fish schooling. PSO is a computational method that optimises a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality (given by a fitness function). It solves a problem by having a population (called swarm) of candidate solutions (called particles) and moving these particles around in the search-space according to simple mathematical formulae (fitness function) over the particle's position and velocity. This is expected to move the swarm toward the regions of the search-space that may contain good candidate solutions. In PSO, the particles "fly" through the problem search space by "following" the best particle of the swarm that currently represent the best solution. Each particle's movement is influenced by its locally best known position (named as *pbest*), but is also guided toward the best globally known positions (named as *gbest*) of the entire population. PSO does not apply evolution operators (mutation or crossover) between individuals of the population as it occurs in genetic algorithms. Instead, in each iteration, all the particles individually update themselves to improve the solutions that they represent. The PSO procedure is described in Algorithm 1 and discussed in Section 3.4.1.

The particles are assessed by a fitness function, which calculates a fitness value to evaluate the quality of the solutions given by them. Iteratively, PSO is guided to select the particle that contains the minimum (or the maximum) fitness value of the entire population, and set this particle as the best candidate solution to the problem. As the position of the best particle represents a region that potentially contains good candidate solutions, PSO attempts to move the swarm toward this region. This process is repeated until the stop condition is reached, moment when the PSO returns the best solution achieved. Figure 4.6 shows the snapshots of the particles' positions on the search-space when the PSO procedure was employed to find the global minimum of the function $f(x, y) = xe^{-x^2-y^2}$. The fitness function employed is this example was $z = f(x, y)$ and the PSO was set up to minimise z. Each particle is configured to be a pair (x, y), which represents a candidate solution for the minimisation of z. The aim of PSO is to find, at each iteration, a particle ρ such that $f(\rho) \leq f(q)$ for all particles q in the swarm. In other words, in the end of each iteration, ρ is the particle that represents the best solution achieved so far. This simple example was set up with 49 particles that were initially scattered according to Figure 4.6a. In a few iterations, PSO was able to find the solution to the problem since most of the particles moved toward the region around the point at $(-\frac{1}{\sqrt{2}}, 0)$, as shown in



(a) Initial particles' position (b) After 5 iterations (c) After 10 iterations

(d) After 15 iterations (e) After 20 iterations (f) After 30 iterations

Figure 4.6: Snapshots of the particles' positions on the search-space to find the global minimum of the function $f(x, y) = xe^{-x^2-y^2}$ using the particle swarm optimisation. Snapshots taken during the 1st, 5th, 10th, 15th, 20th, 30th iterations of the PSO procedure. The fitness function employed in this example was $z = f(x, y)$ and the PSO was set up to minimise z. Forty nine particles was created (red circles) and each one represents a pair (x, y) as a candidate solution to the problem. All particles represents the PSO's swarm.

Figures 4.6b–4.6f. The solution of the minimisation of z is $-\frac{1}{\sqrt{2e}}$ at $(-\frac{1}{\sqrt{2}}, 0)$.

PSO is a popular tool due to its simplicity and its effectiveness in solving a wide range of NP-Complete problems at a low computational cost [90]. However, meta-heuristics such as PSO do not guarantee an optimal solution is ever found. Therefore, the PSO technique makes few or no assumptions about the problem being optimised, and thus this technique does not guarantee the bounds for the optimality of the achieved solution. An extensive survey of PSO applications can be found in [65, 103, 104, 113], including the use of PSO in scheduling problems. Recently, a comprehensive review on theoretical and experimental works on PSO has been published in [37].

4.4.2 Particle as a Candidate Schedule Solution

In PSO, the swarm is composed of a number of particles that encompass the solution search-space. To define a particle for the context of the scheduling of workflow ensemble, let $\mathcal{E} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$ be a workflow (DAG) ensemble and $\mathcal{R}$ be a set of resources. A particle is represented by an array of length k, where k is the total number of tasks contained in the ensemble $\mathcal{E}$. The value assigned to each particle's dimension (array's position) is an integer that represents the index of a computing resource in $\mathcal{R}$. Therefore, each particle of the swarm represents a candidate schedule that simply allocates (mapping) $\mathcal{E}$ to $\mathcal{R}$. A k-dimensional particle is illustrated in Figure 4.7. The schedule represented by the

particle is evaluated by the fitness function (scheduler's objective function), which tells whether the schedule is a good solution as well as satisfies the desired qualities of service, such as meet the deadline.

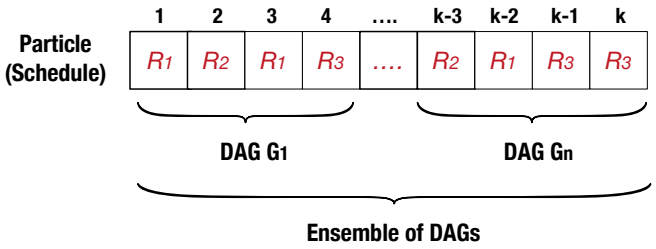


Figure 4.7: A sample of a particle that represents a schedule for an ensemble of n inter-related DAGs, where $\mathcal{R} = \{R_1, R_2, R_3\}$ is a set of resources.

4.4.3 Scheduling Policy

Workflows in an ensemble may differ not only in their parameters, but also in their priority. For example, in CyberShake some geographic locations may be in heavily populated areas or have sensitive facilities such as power plants, while others may be less important. Scientists typically prioritise the workflows in such an ensemble so that important workflows are finished first [97]. This enables them to see critical results early, and helps them to choose the most important workflows when the time and financial resources available for computing are limited. Due to this, the DAGs represented by the particle are scheduled independently by the scheduler on a "first come, first served" basis. We assume that each DAG in the ensemble is given a numeric priority which indicates how important the DAG to the user is. The highest priority DAG is the first one to be scheduled. Therefore, the merging of DAGs into a one single-DAG (Figure 4.4) is not considered. Regarding the node (task) ordering within a DAG, this follows the topological order given by the depth-first search algorithm. Other types of node ordering could be used instead, such as the upward rank employed by the HEFT algorithm in [130, 149].

As the DAGs are scheduled independently, one after another, the scheduler uses a gap search or backfilling technique to schedule tasks on resources [33]. Basically, this technique searches for free spaces between tasks already scheduled, and thus a task from a DAG that is being scheduled can be executed before tasks already scheduled. This method takes advantage of the gaps left on resources between scheduled tasks as a result of communication dependencies (Figure 4.5). For example, by using idle times left on resources to schedule tasks from other DAGs, the scheduler can achieve schedules that may result in the reduction of the overall makespan and costs involved with the leasing of VMs, besides to be able to promote the maximisation of resources utilisation.

4.4.4 Fitness Function as a Scheduler's Objective Function

Let $f : \mathbb{N}^k \rightarrow \mathbb{R}$ be the fitness function that must be minimised. This function takes a k-dimensional particle as an argument in the form of an array of integers, and produces a

real number as output, which indicates the particle's fitness value. The aim of PSO is to find, at each iteration, a particle ρ such that $f(\rho) \leq f(q)$ for all particles q in the swarm. The maximisation can be performed by considering the function $f' = -f$ instead. Importantly, the particle only informs on which resource of $\mathcal{R}$ each task of $\mathcal{E}$ is scheduled to be run. Complying with the scheduler's policies, it is the role of the fitness function in calculating the timing of scheduling events. To show the scheduler's flexibility, three different objective functions were employed: minimisation of overall makespan, minimisation of costs, and maximisation of fairness. Except for the objective function of maximising the fairness, which was given two fitness functions, the other objective functions were given only one. An overview of how these fitness functions are employed in the PSO procedure is illustrated in Algorithm 11.

Algorithm 11 Fitness Function Overview

1: **procedure** FITNESS FUNCTION (a particle $\rho[\,]$)
2: Calculates the fitness value for ρ according to the specification designated for this function
3: **return** the fitness value of ρ to Algorithm 1
4: **end procedure**

Proposed scheduler as a Makespan-aware scheduler To comply with the minimization of the schedule makespan (application execution time), the fitness function f_1 is employed. This function produces as output numbers associated with the overall makespan of a given particle (schedule) ρ. By using this function, the PSO is setup to minimise the resolution of the problem, since its goal is to find particles that represent schedules with the lowest overall makespan. The fitness function f_1 is defined as follows:

$$f_1(\rho) = \mu(\rho) \tag{4.1}$$

where $\mu(\rho)$ is the overall makespan of the schedule given by the k-dimensional particle ρ, that is, the start time of the first task of the first DAG in the ensemble $\mathcal{E}$ to the finish time of the last task of the last DAG in the ensemble $\mathcal{E}$.

Proposed scheduler as a Fairness-aware scheduler When scheduling several DAGs simultaneously on the same set of resources, each DAG may experience a *slowdown* in its execution, when compared to the runtime of the same DAG if it had the resources on its own. A schedule is considered fair when each DAG of the ensemble experiences equal (or similar) slowdown in its execution. The slowdown value of a DAG $i \in \mathcal{E}$, $\sigma(i)$, in the schedule given by the particle ρ is defined as in [149]:

$$\sigma_i = \frac{\mu_i^{own}(\rho)}{\mu_i(\rho)}, \tag{4.2}$$

where μ_i^{own} is the makespan of a DAG i if it had the available resources on its own, and μ_i is the makespan of a DAG i when sharing resources along with all other DAGs. It is expected that slowdown values are in the range $[0, 1]$, with values closer to 1 indicating a small slowdown. To maximise fairness, two fitness functions g_1 and g_2 were employed. The function g_1 is defined as the ratio between the lowest (minimum) slowdown and the

peak (maximum) slowdown experienced by the DAGs in the schedule. PSO is configured to maximise the resolution of the problem since its goal is to find particles that represent high $h(\rho)$ values, i.e., fairness schedules.

Derived from [11], the function g_1 is defined as the ratio between the lowest (minimum) slowdown and the peak (maximum) slowdown experienced by the DAGs in the schedule. In this case, the PSO is configured to maximise the resolution of the problem since its goal is to find particles with higher $g_1(\rho)$ values, because the maximum expected slowdown value for a DAG is 1. The function g_1 is defined as:

$$g_1(\rho) = \frac{\min\limits_{\forall i \in \mathcal{E}}(\sigma_i)}{\max\limits_{\forall i \in \mathcal{E}}(\sigma_i)} \tag{4.3}$$

Derived from [149], the fitness function g_2, is defined on the basis of the absolute value of the difference between the slowdown of each DAG and the average slowdown of all DAGs in the ensemble $\mathcal{E}$. The function g_2 is defined as:

$$g_2(\rho) = \frac{1}{|\mathcal{E}|} \sum_{\forall i \in \mathcal{E}} \left|\sigma_i - \bar{\sigma}\right|, \tag{4.4}$$

where $\bar{\sigma}$ is the average slowdown value for all DAGs in $\mathcal{E}$ given by:

$$\bar{\sigma} = \frac{1}{|\mathcal{E}|} \sum_{\forall i \in \mathcal{E}} \sigma_i \tag{4.5}$$

A low g_2 value indicates schedules where slowdown difference between DAGs is small, and therefore these schedules are reasonably fair to each DAG. Therefore, the use of the fitness function g_2 demands that the PSO be configured to minimise the resolution of the problem since its goal is to find particles that result in smaller g_2 values.

Proposed scheduler as a Cost-aware scheduler In cloud scenarios, scheduling of workflow ensembles has to take into account not only performance-related metrics such as makespan or fairness, but also financial expense, since the resources obtained from commercial clouds have monetary costs associated with them. To minimise costs, we employ the fitness function h_1. This function is defined on the basis of the monetary execution cost of a schedule represented by a particle ρ. PSO minimises the resolution of the problem in this case, since its goal is to achieve particles that represent low-cost schedules. The function h_1 is defined as follows:

$$h_1(\rho) = \mathcal{C}(\rho), \tag{4.6}$$

where $\mathcal{C}(\rho)$ is the overall monetary execution cost of the schedule given by ρ, which is defined as follows:

$$\mathcal{C}(\rho) = \sum_{\forall r \in \mathcal{R}} \left(c_r \cdot \lceil t_{r,\rho} \rceil\right) \tag{4.7}$$

where c_r is the cost per time unit for using the resource $r \in \mathcal{R}$, and $t_{r,\rho}$ is the total time units of usage of r in the schedule given by ρ, where partial units are rounded up.

4.4.5 Summary

This section presented the proposed PSO-based flexible scheduler for workflow ensembles and four different fitness functions that work as the scheduler's objective function. The feature of flexibility is accomplished by the possibility of developing a specific fitness function that addresses the user's needs. In order words, the flexibility of the scheduler is accomplished by modelling the scheduling problem as a PSO and developing a fitness function (module) that acts as an objective function of the scheduler. By having a population of candidate schedule solutions, PSO uses a fitness function to select the best solution of the entire population. Therefore, by having different fitness functions, PSO can select the best solution in different ways as well as with different types of constraints. To highlight this flexibility, four different (conflicting) fitness functions were developed with respect to the minimisation of overall makespan, maximisation of fairness, and minimisation of costs. A summary of these functions is shown in Table 4.2.

Table 4.2: Summary of fitness functions as the scheduler's objective function

Proposed scheduler as	Fitness function	Description of the fitness value produced by the function
Makespan-aware scheduler	f_1	Overall makespan of the ensemble
Fairness-aware scheduler	g_1	Ratio between the minimum and the maximum slowdowns
	g_2	Absolute difference between slowdowns and the average slowdown
Cost-aware scheduler	h_1	Monetary execution cost of the ensemble

4.5 Evaluation Methodology

This section describes the methodology of the evaluation of the efficacy of the PSO-based flexible scheduler for workflow ensembles. The objective of this evaluation is to assess the flexibility of the scheduler in cope several types of objective functions when scheduling workflow ensembles on cloud resources, such as minimise the overall makespan, maximise fairness, and minimise costs. Thus, the objective of the evaluation is to show the versatility of the proposed scheduler that is adaptable to meet several scenarios of the scheduling problem, since we designed the scheduler to be able to accept different objective functions. Therefore, this section proposes to highlight the scheduler's proposed flexibility by accepting many objective functions of different types and shows how the scheduler faces the same set of workflow ensembles differently because ordinary workflow schedulers are not able to cope.

4.5.1 Applications as Workflow Ensembles

In order to evaluate the proposed flexible scheduler on a set of workflows, we created ensembles using workflows available from the workflow generator gallery[14]. The gallery

is composed of synthetic workflows modelled using patterns taken from real applications [129], and the evaluation was carried out using the following applications: Epigenomics, LIGO, Montage, and SIPHT. It was considered only workflows with 50 tasks (DAG nodes) since the focus of the experiment was to evaluate the flexibility of the proposed scheduler. For each workflow application, 100 different workflows were generated differing in the weights of the DAG nodes (computational demands) and DAG edges (communication demands). The corresponding collection of synthetic workflows contains 4 types of workflow applications, 1 workflow size (number of tasks), and 100 workflows per application and size, summarising in a total of 400 workflows. Using this collection of synthetic workflows, we constructed ensembles for each type of workflow application. The number of workflows in an ensemble depends on the particular application, but, in this evaluation, the size of the ensemble ranged from 1 to 10 workflows. Given an ensemble size and the type of the application, the ensemble is therefore constructed by choosing randomly workflows from this collection of workflows regarding this application type

4.5.2 Cloud Scenario

The resource scenarios to execute the workflow are shown in Table 4.5. They employ resources based on the configurations of the Amazon's EC2. The cloud scenario of the evaluation contains three types of virtual machines (small, large, x-large), resulting in a total of 12 virtual machines. The availability of bandwidth of 100 Mbps was considered in the communication channels that connect the resources inside the provider. We assume that the resources are exclusively used for the workflow ensemble execution, as described in Section 4.3.

Table 4.3: Cloud scenario of the experimental evaluation of the proposed PSO-based flexible scheduler for workflow ensembles.

Type	Cores	Performance per core	Price per unit of time	Quantity
Small	2	2	$ 0.026	4
Large	2	6.5	$ 0.120	4
X-Large	4	13	$ 0.239	4

4.5.3 Simulator/Calculator

Figure 4.8 shows an overview of how the evaluations were carried out. The data inputs for the scheduler are DAX files (ensemble) and a VMs file. Each DAX file contains the description of the workflow in XML format, while the VMs file contains information about the resources from Table 4.5. DAX is a file format used by the Pegasus Workflow Management System[12] to describe a workflow application. After the schedule is produced by the scheduler, it is used as an input to the simulator. The simulator simulates the ensemble execution given by the schedule and calculates some metric values such as: the overall makespan, slowdown values, monetary execution costs and speedup. Importantly,

[12] `https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator`

at simulation time, all tasks of all workflows are re-ranked according to the execution finish time estimated by the produced schedule. For more details regarding the simulator, please see Appendix A.

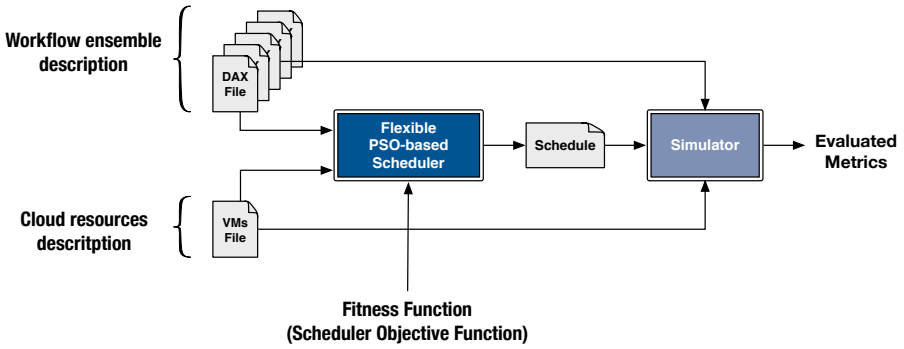


Figure 4.8: Overview of the experimental evaluation of the proposed PSO-based flexible scheduler for workflow ensembles.

4.5.4 Experimental PSO Parameters

We used the JSwarm-PSO package[13] to conduct the experimental evaluation of the proposed PSO-based flexible scheduler. The number of particles was set to 500 and the number of iterations was set to 50. Following the default JSwarm-PSO's configuration, the particle's inertia ω was set to 1.5, and it decreases 1% at each iteration. The particle's acceleration values a_1 and a_2 were set to 0.95, and both decrease 1% at each iteration.

4.5.5 Evaluated Metrics

The evaluated metrics are the overall makespan of the workflow ensemble, the monetary cost of the schedule, and the runtime of the scheduler to find a solution. To measure fairness, we employ the Jain's fairness index [83], an index that was developed for resource sharing or allocation problem. Consider a set of values $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, the Jain's fairness index for the set $\mathcal{X}$ is defined as follows:

$$\mathcal{J}(x_1, x_2 \ldots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n\left(\sum_{i=1}^{n} x_i^2\right)} \tag{4.8}$$

The Jain's fairness index ranges between $\frac{1}{n}$ (worst case, unfair) and 1 (best case, fair). An index value closer to 1 indicates that the slowdown difference between the DAGs is small, and thus the schedule is reasonably fair to each DAG.

The main advantage of sharing multiple resources is to attempt to speed up the execution of the application by exploring the gaps (idle times) left on resources between dependent-tasks already scheduled. To compare the performance of the resource sharing among the workflows, the speedup was employed. Given a schedule for a workflow ensemble, we define speedup as the ratio between the overall makespan taken to process the

[13] `http://jswarm-pso.sourceforge.net/`

workflows sequentially without sharing the resources, and the overall makespan taken to process the same set of workflows but with the sharing of resources among the workflows. The speedup value of a schedule (particle) ρ for an ensemble $\mathcal{E}$ is calculated as follows:

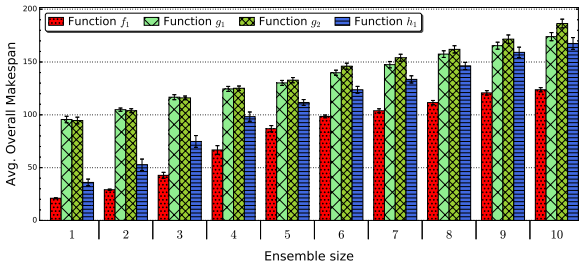
$$Speedup\,(\rho) = \frac{1}{\mu(\rho)} \sum_{\forall i \in \mathcal{E}} \mu_i^{own}(\rho), \tag{4.9}$$

where $\mu_i^{own}(\rho)$ is defined in Equation 4.2.

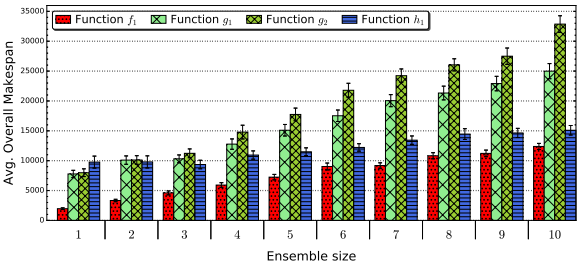
Speedup values grater than 1 are expected. Otherwise, resource sharing among the workflows is ineffective to speedup the execution of the ensemble.

4.6 Results

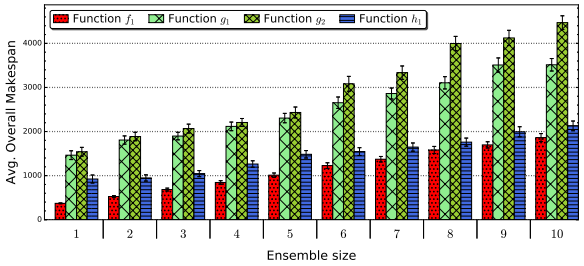
This section presents numerical results that were produced to assess the flexibility of the proposed PSO-based scheduler for workflow ensembles. To evaluate this flexibility feature, three types of conflicting objective functions were employed (Table 4.5): minimisation of the overall makespan, maximisation of fairness, and minimisation of costs. These functions make the proposed scheduler into a makespan-aware, fairness-aware, and cost-aware scheduler, respectively. The graphics showed in this section were generated over 100 simulations and the data were presented with 95% of confidence interval.



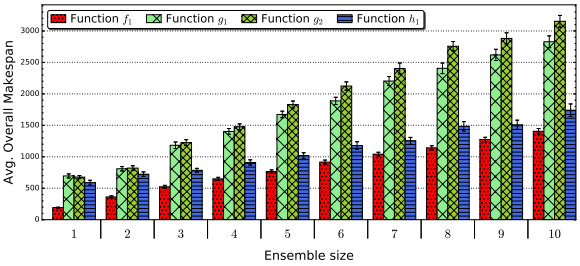
(a) Montage Application



(b) Epigenomics Application



(c) SIPHT Application



(d) LIGO Application

Figure 4.9: Average overall makespan for the Montage, Epigenomics, SIPHT, and LIGO applications by using the proposed scheduler as a makespan-aware scheduler. The x-axis represents the number of workflows within the ensemble, which ranges from 1 to 10.

4.6.1 Scheduler as a Makespan-aware Scheduler

Figure 4.9 shows the average overall makespan of the evaluation results for the Montage, Epigenomics, SIPHT, and LIGO applications. As it was expected, the employment of the fitness function f_1 drove the proposed scheduler in producing schedules that resulted in lower average overall makespans for all applications and ensemble sizes than that of the other three fitness functions g_1, g_2, and h_1. For example, to produce schedules for the Montage application as ensembles of 10-workflows using the function f_1, the average values of the overall makespan were approximately 24%, 31%, and 7% lower than the average values of the solutions achieved by the employment of the functions g_1, g_2, and h_1, respectively. Also, these reductions were approximately 43%, 55%, and 18% for the Epigenomics ensembles; 45%, 58%, and 9% for the SIPHT ensembles; and 37%, 39%, and 4% for the LIGO ensembles, respectively. In general, using a dedicated fitness function to comply with the objective of minimising the overall makespan makes the proposed PSO-based flexible scheduler in acting as a makespan-aware scheduler, achieving schedules with reduced overall makespan for all experimental cases.

The achievement of schedules with reduced makespan was possible because the fitness function f_1 is defined to setup the particle's fitness values as the overall makespan of the workflow ensemble. As the PSO is setup to minimise the resolution of the problem, it picks at each iteration the particle that represents the best schedule achieved so far because it results in the lowest overall makespan. Therefore, we can observe a better performance in reducing the overall makespan is through the use of the fitness function f_1 than that of the other fitness functions g_1, g_2, and h_1.

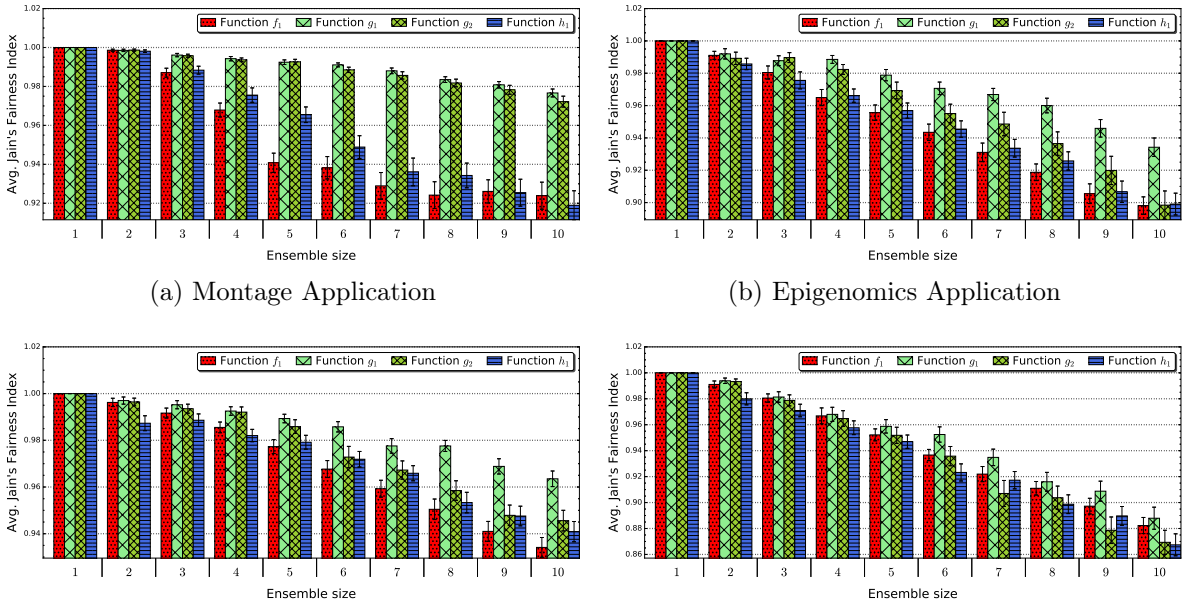
Figure 4.9 reveals that the average overall makespan increases linearly with the length of the ensemble for all applications evaluated. It was an expected behaviour since the greater the number of DAGs in the ensemble (application), the larger the overall makespan of schedules. However, f_1 was the only function that managed to curb the overall makespan increase as the ensemble size increases. The f_1 function was the only one that contributes to make the proposed scheduler into a makespan-aware scheduler. Figure 4.9 also reveals that the use of the fitness functions g_1 and g_2, which attempt maximising the fairness, drove the PSO in produced poor schedules in the matter of minimising the overall makespan since both g_1 and g_2 resulted in the largest overall makespan for all cases. As the next section shows, the goal of minimising the overall makespan given by f_1 drove the PSO in producing unfair schedules.

4.6.2 Scheduler as a Fairness-aware Scheduler

This section presents the results given by the employment of the functions g_1 and g_2, where both functions try to make the proposed scheduler into a fairness-aware scheduler. Figure 4.10 presents the average Jain's fairness index of the experiments for the Montage, Epigenomics, SIPHT, and LIGO applications when the proposed scheduler was used as a fairness-aware scheduler. As it was also expected, the fitness functions g_1 and g_2 were able to drive PSO in achieving fairer schedules for all applications and ensemble sizes, when compared to the schedules produced using the functions f_1 (minimise the overall makespan) and h_1 (minimise the costs). In fact, for all cases, the function g_1 could manage

fairer solutions than g_2. For example, using g_1 to conduct the production of schedules for the Montage application as ensembles of 10-workflows, the averages Jain's indexes were approximately 1%, 10%, and 8% higher than that produced by the others functions g_2, f_1, and h_1, respectively. Also, these increases were approximately 4%, 7%, and 5% for the Epigenomics 10-workflow ensembles; 5%, 3%, and 4% for the SIPHT 10-workflow ensembles; and 4%, 5%, and 4% for the LIGO 10-workflow ensembles, respectively. Indeed, the use of two different fitness functions to make the scheduler in producing fair schedules in two different ways highlights the flexibility of the proposed scheduler in becoming a scheduler aware of *whatever-you-want*.

A Jain's index value closer to 1 indicates that the slowdown difference among the DAGs in an ensemble is small, and therefore the schedule is reasonably fair to each DAG. In fact, for most of the cases, the fitness function g_1 and g_2 achieved averages higher than 0.9. However, the approach of trying to reduce the difference between the minimum and maximum slowdown values (g_1) seems to be more effective in maximise fairness than the approach of trying to reduce the average absolute difference between slowdown values and the average slowdown (g_2). Figure 4.10 reveals that the Average Jain's fairness indexes gradually decreases linearly with the length of the ensemble for all applications evaluated. It was also an expected behaviour since the greater the number of DAGs in the ensemble, the greater the competition among the DAGs to use the resources, the smaller the Jain's index value.



(a) Montage Application

(b) Epigenomics Application

(c) SIPHT

(d) LIGO Application

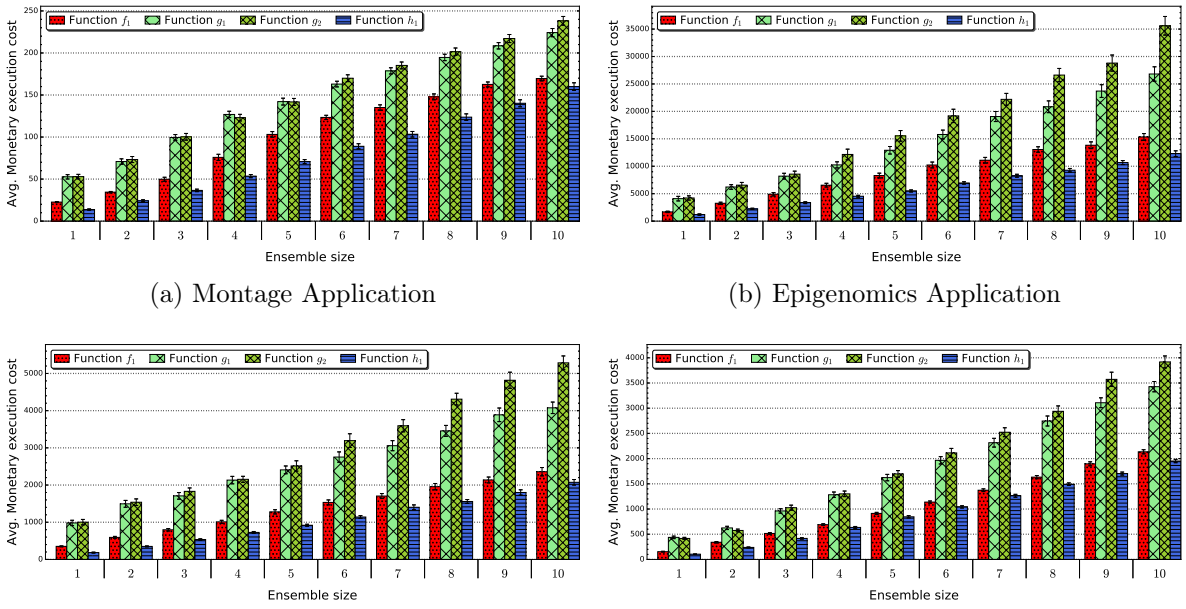
Figure 4.10: Average Jain's fairness indexes for the Montage, Epigenomics, SIPHT, and LIGO applications by using the proposed scheduler as a fairness-aware scheduler. The x-axis represents the number of workflows within the ensemble, which ranges from 1 to 10.

Figure 4.10 also reveals that using g_1 and g_2 underwent in less fairness degradation than did the f_1 and h_1 as the number of DAGs in the ensemble increases. For example, in the comparison among the ensembles with 1 and 10 Montage DAGs, the averages

Jain's indexes were reduced by approximately 7.75%, 2%, 2.5%, and 8% when the fitness functions f_1, g_1, g_2, and h_1 were employed in the scheduler, respectively. The same pattern is observed for the others applications. The fairness degradation among ensembles of 1 and 10 DAGs were 10%, 6.25%, 8%, and 8% for the Epigenomics application; 5.5%, 3%, 5.5%, and 6% for the SIPHT application; and 12%, 10.5%, 13%, and 13% for the LIGO application. We can observe that using a proper fitness function to share the resources equally among the workflows is demanded when it is necessity to produce fair schedules.

The g_1 and g_2 are only two simple examples of fitness functions to conduct the PSO in maximising fairness, i.e., to make the proposed scheduler into a fairness-aware scheduler. However, other fairness functions could be used instead since the proposed PSO-based scheduler allows this flexibility. For example, by setting up the PSO to minimise the resolution of the problem, a fitness function that returns the standard deviation of the slowdown values could be used since a standard deviation close to 0 indicates that the slowdown values tend to be close to the average slowdown. This example may also be a good fitness function to make the proposed scheduler into a fairness-aware. This example along with the other functions (f_1, g_1, g_2, and h_1) emphasises the flexibility of the scheduler in producing fair schedules in accordance the user's needs.

The objective of maximising fairness involves a drawback with respect to financial costs. Due to these conflicting objectives, a fair schedule implies in a higher overall makespan, as shown in Figure 4.9, which causes an increase in costs because the cloud resources have monetary costs associated with them. In the next section, we discuss the results given by the function h_1, which attempts to drive PSO in producing low-cost schedules, making the proposed flexible scheduler into a cost-aware scheduler.



(a) Montage Application

(b) Epigenomics Application

(c) SIPHT

(d) LIGO Application

Figure 4.11: Average monetary execution costs for the Montage, Epigenomics, SIPHT, and LIGO applications by using the proposed scheduler as a cost-aware scheduler. The x-axis represents the number of workflows within the ensemble, which ranges from 1 to 10.

4.6.3 Scheduler as a Cost-aware Scheduler

In addition to minimising the makespan and maximising fairness, the proposed flexible scheduler can also be applied to minimising the monetary execution costs of workflow ensembles scheduling on cloud resources. Figure 4.11 presents the average monetary execution costs of the experiments for the Montage, Epigenomics, SIPHT, and LIGO applications when the proposed scheduler was used as a cost-aware scheduler. This experiment was carried out using the resources described in Table 4.5. In this evaluation, it was expected the fitness function h_1 would conduct the PSO algorithm in producing cheaper schedules, and indeed this expectation has been achieved for all applications and ensemble sizes. For example, employing the function h_1 to make the proposed scheduler into a cost-aware scheduler to schedule the Montage applications as ensembles of 8-workflows, it was able to achieve schedules approximately 17%, 37%, and 39% cheaper on average than the schedules achieved by the scheduler using the functions f_1, g_1, and g_2, respectively. Moreover, these cost reductions were approximately 29%, 55%, and 65% for the Epigenomics application; 15%, 46%, and 61% for the SIPHT application; and 9%, 46%, and 49% for the LIGO application, respectively.

Figure 4.11 mirrors Figure 4.9, where the average monetary execution costs also increase linearly with the length of the ensemble for all applications evaluated. However, it was also an expected behaviour since the greater the number of DAGs in the ensemble (application), the larger the overall makespan of schedules, and as a consequence, the more expensive the application execution. However, h_1 was the only function that managed to curb the cost increase as the ensemble size increases, making the proposed scheduler into a cost-aware scheduler straight away. In fact, the fitness function h_1 has complied with its duty to conduct the PSO to achieve cost-efficient schedules for scientific applications expressed as workflow ensembles. Besides that, using the function h_1 instead of g_1 and g_2 is just an example of how flexible it is to drive the scheduler from a scheduling fairness maximisation problem to a scheduling cost minimisation problem.

Table 4.4: Average speedups of the schedules produced for the Montage, Epigenomics, SIPHT and LIGO applications using the proposed PSO-based flexible scheduler.

	Fitness	Ensemble Size									
	Function	1	2	3	4	5	6	7	8	9	10
Montage	f_1	1.0	1.9	2.3	2.5	3.0	3.7	4.2	4.8	5.5	6.0
	g_1	1.0	1.7	2.3	2.8	3.2	3.7	4.2	5.9	4.8	5.2
	g_2	1.0	1.7	2.2	2.7	3.2	3.5	3.9	2.4	4.5	4.6
	h_1	1.0	1.6	2.0	2.3	2.8	3.2	3.6	4.1	4.4	4.7
Epigeno.	f_1	1.0	1.6	2.2	2.7	3.0	3.2	3.5	3.6	3.9	4.0
	g_1	1.0	1.5	2.0	2.3	2.5	2.5	2.7	2.7	2.9	2.9
	g_2	1.0	1.5	2.0	2.1	2.2	2.3	2.3	2.3	2.5	2.5
	h_1	1.0	1.4	1.8	2.2	2.5	2.8	3.0	3.1	3.3	3.6
SIPHT	f_1	1.0	1.7	2.1	2.5	3.1	3.2	3.2	3.4	4.0	3.9
	g_1	1.0	1.6	2.2	2.1	2.2	2.6	2.5	2.6	3.0	3.0
	g_2	1.0	1.5	2.1	2.1	2.3	2.3	2.4	2.4	2.5	2.5
	h_1	1.0	1.5	1.8	2.0	2.3	2.6	3.2	3.2	3.3	3.6
LIGO	f_1	1.0	1.7	2.2	2.6	2.8	2.9	3.1	3.2	3.3	3.3
	g_1	1.0	1.6	1.7	1.9	2.0	2.1	2.1	2.2	2.2	2.3
	g_2	1.0	1.6	1.7	1.8	1.9	1.9	2.0	1.9	2.1	2.1
	h_1	1.0	1.5	2.0	2.2	2.4	2.6	2.7	2.8	3.0	3.0

4.6.4 Speedup Analysis

To speedup the execution time (overall makespan) of workflow ensembles, the resource sharing among the workflows will eventually have to occur. To evaluate this, the speedup was calculated and it is shown in Table 4.4. For all applications and ensemble sizes, the proposed scheduler was able to produce schedules that resulted in speedups greater than 1, meaning that the resource sharing among the workflows has been achieved. This means that the scheduler was capable to take advantage of the free gaps left on the resources to schedule tasks from other workflows in order to reduce the application execution time (makespan) as well as to maximise the resource utilisation.

Section 4.5.5 defines speedup as the ratio between the time taken to process sequentially the workflows of an application (ensemble) and the time to process the same set of workflows in a non-sequential fashion. Therefore, the only function that directly impacts speedup is the function f_1 since it attempts to minimise the overall makespan. Due to this, it was expected f_1 to drive PSO in producing higher speedups than the other fitness functions, and indeed this expectation has been achieved for all applications and ensemble sizes. For example, the function f_1 was able to produce higher speedup values than those produced by the functions g_1, g_2, and h_1 for the Montage application as 10-workflows ensembles. In this case, the f_1 achieved an average speedup approximately of 6.0 against 5.2 , 4.6 and 4.7 produced by the functions g_1, g_2, and h_1, respectively. This observation is mirrored for the Epigenomics, SIPHT, and LIGO applications as ensembles of 10-workflows. For example, the average speedups achieved by using f_1, g_1, g_2, and h_1 were approximately of 4.9, 2.9, 2.5, 3.6, respectively for the Epigenomics; 3.9, 3.0, 2.5, 3.6, respectively for the SIPHT; and 3.3, 2.3, 2.1, 3.0, respectively for the LIGO. Regardless of the fitness function employed in the scheduler, it can be observed from Table 4.4 that the scheduler was able to achieve speedups greater than 1 for all cases when there was more than one workflow in the ensemble. Indeed, this shows that the scheduler was able to fill the empty gaps left on resources (as illustrated, for instance, in Figure 4.5) between scheduled tasks of previous workflows as a result of their communication dependencies.

4.6.5 Runtime of the Proposed Flexible Scheduler

The average runtime to execute the proposed PSO-based flexible scheduler is shown in Figure 4.12. These averaged data is over 100 runs to schedule workflow ensembles from Montage, Epigenomics, SIPHT, and LIGO applications with four different fitness functions (f_1, g_1, g_2, and h_1). The running time to produce schedules appears to grow slightly exponentially with the number of workflows (DAGs) in the ensemble considered. The runtime of the scheduler using g_1, g_2 are slightly slower since both functions need to make an additional makespan computation for each a DAG of the ensemble. In other words, unlike f_1 and h_1, the functions g_1, g_2 need to compute the makespan when a DAG is and is not sharing the resources among other DAGs in order to calculate the slowdown values.

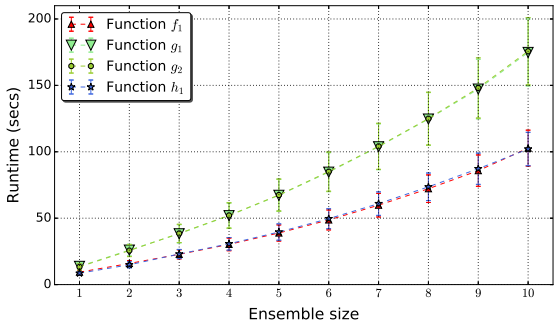


Figure 4.12: Average runtime of the proposed PSO-based scheduler over 100 runs by using workflow ensembles from Montage, Epigenomics and LIGO applications with four different fitness functions. The x-axis represents the number of workflow within the ensemble, which ranges from 1 to 10.

4.6.6 Discussion

Minimisation of makespan, maximisation of fairness, and minimisation of cost are indeed the most common objective functions for a workflow scheduler in the literature [118, 140, 146]. Schedulers are indeed mostly designed to optimise only one objective function [33,36, 47,97,110,145,149], while a few of them are developed to address multi-objective functions (in general two functions) at the same time [61,62]. However, none of them are designed to allow the replacement of these objective functions, specially schedulers that cope several inter-related workflows grouped into ensembles. To the best of our knowledge, there is no study in the literature that proposes a scheduler that facilitates the replacement of the objective function according to the user's needs. The proposed PSO-based scheduler is a novel scheduler for workflow ensembles that provides this flexibility. Normally, for each objective established for the scheduling procedure of an application, a different scheduler is often used to achieve such goal. For example, a specific makespan-aware scheduler is picked by the user when he/she demands to achieve the objective of minimisation of the application's makespan. Similarly, the user picks a fairness-aware scheduler to achieve the maximisation of fairness and a cost-aware for the minimisation of cost. The flexibility of replacing the objective function is not common to occur in current schedulers because these functions are usually tightly attached to the scheduling algorithm. Our PSO-based scheduler, however, provides this flexibility of replacing the objective functions, which makes our scheduler into a scheduler aware of the user's need.

As a general contribution of the presented results, we are able to conclude from these simulations that the feature of flexibility in replacing the scheduler objective function extends the usefulness of the workflow scheduler to be applied in various scenarios. For a same set of synthetic workflow employed in the evaluation, for instance, the fitness function f_1 was able to drive PSO in achieving schedules with smallest overall makespan for all cases, while fairest schedules can only be able to be found using the functions g_1 and g_2. However, the function h_1 was the only function of the four available that was more suitable to achieve cheapest schedules. Note that these functions are only sample functions that were used to highlight the flexibility given by the proposed scheduler. We claim that the functions f_1, g_1, g_2, and h_1 were only sample functions used to highlight

the scheduler's flexibility feature and that other more sophisticated functions could be used instead to attempt to achieve better results.

Therefore, in the presented evaluation, the proposed scheduler was firstly turned into a makespan-aware scheduler, then in a fairness-aware scheduler, and lastly into a cost-aware scheduler. In other words, in the presented evaluation, the proposed scheduler was simply turned into three different completely schedulers thanks to the employment of the particle swarm optimisation (PSO). The proposed scheduler is accomplished by modelling the workflow scheduling problem as a PSO and developing a fitness function that acts as the scheduler's objective function. Due to the possibility to create as many fitness functions as desired, the scheduler is considered flexible since it allows the replacement of the objective function according to the user's needs, expanding the usability of the scheduler to be adopted across multiple cloud scenarios.

Therefore, the proposed flexible scheduler provides a freedom to the user to be able to design different objective functions to manage the production of schedules in different ways and according to their needs. For example, besides the scheduling scenarios shown in this study, the user may need to develop a fitness function that drives PSO in achieving schedule solutions that minimise the ratio between the cost and the overall makespan, or even minimise costs while complying with deadline/storage/energy constraints. Note that, in addition to being flexible, the proposed scheduler is also ready to schedule a set of inter-related workflows grouped into ensemble simultaneously on a set of cloud resources. Indeed, the problem of scheduling a set of workflows simultaneously has been neglected by most studies in scheduling, specially those studies using cloud computing.

The proposed scheduler is flexible, however, it does not include a mechanism (or a procedure) to produce robust schedules under uncertainties of available bandwidth. For this purpose, it was employed the proactive procedure presented in Chapter 3. The next section presents an assessment of the flexible scheduler using this procedure.

4.7 Robustifying the PSO-Based Scheduler for Workflow Ensembles

The PSO-based workflow scheduler presented in the previous section is not designed to address inaccurate information about the available bandwidth stemming from using monitoring/measurement networking tools. Due to this, poor schedules are expected to be produced when the measured bandwidth estimates used at scheduling time do not correspond to the actual ones experienced at the execution time. In an attempt to drive the PSO-based scheduler in producing robust schedules in the face of such uncertainty, an adapted version of the proactive procedure presented in Chapter 3 is employed here.

Imprecise input data imposes special challenges in the scheduling of a single workflow, as it was intensively discussed in Chapter 3. This challenges, however, becomes even more complex when it involves the scheduling of several workflows simultaneously. In this case, neglecting the inaccuracy of input information in the production of schedules will not affect the execution of just one workflow but the execution of several workflows. The procedure presented in Chapter 3 has showed to be effective in improving the robustness of

deterministic schedulers of single-workflows under the uncertainty of available bandwidth in intra-cloud and inter-cloud links of a hybrid cloud. Thus, motivated by the results in Chapter 3, this chapter introduces the RobWE scheduler, a scheduler that extends the robustness of the PSO-based scheduler for workflow ensembles with the employment of the procedure. The effectiveness of the RobWE scheduler is compared to its non-robust version, i.e., the original PSO-based scheduler for workflow ensembles without using the procedure. Results of simulations considering diverse scenarios, based on several degrees of uncertainty in available bandwidth estimates, characteristics of bandwidth estimations and applications as workflow ensembles, demonstrate the advantages of the RobWE scheduler. The simulation results demonstrate the procedure was also effective in extending the robustness of the scheduling of workflow ensembles in a hybrid cloud.

This chapter is organised as follows. Section 4.7.1 presents the revised version of the procedure in order to satisfy several types of scheduler's objective functions, while Section 4.7.2 describes the RobWE's evaluation methodology in a hybrid cloud. The results are presented in Section 4.7.3, while a discussion regarding the effectiveness of the "robustified" PSO-based scheduler for workflow ensembles is presented in Section 4.7.4.

4.7.1 Procedure to handle the Uncertainty Values of the Available Bandwidth in the Scheduling Production

As discussed in Chapter 3, the estimate of the available bandwidth may deviate from that measured by an expected percent error p during the execution time of an application [16, 19]. In a scenario in which the expected uncertainty of the available bandwidth is $p = 50\%$, for instance, a data transfer that is initially estimated by the scheduler to take 30 seconds, it could take between 15 and 45 seconds during the execution of the application. To quantify such uncertainties, the quality of information model presented in [44] is employed by the procedure. Let b be the estimated value of the available bandwidth and p be the expected uncertainty of b. According to this model, during the execution of the workflow, the estimate of the inter-cloud available bandwidth may experience a variation from $b-p\%$ to $b + p\%$. Most schedulers in the literature neglect the presence of such variability p, especially scheduler that schedules a set of workflows simultaneously (Table 4.1). As a result, the produced schedules may have their performance degraded as a result of large values of p at applications runtime, meaning that the schedules are poorly produced.

As a proactive approach to minimise the negative impact of using imprecise information about the available bandwidth, the procedure in Chapter 3 produces a deflating multiplier value that is applied to the measured available bandwidth before being furnished to the scheduler. According to an expected uncertainty p, the procedure applies a reduction factor $\mathcal{U}$ to the measured inter-cloud bandwidth to prevent inaccurate information from affecting the performance of the execution of the workflow ensemble application. Motivated by the results presented in Sections 3.7 and 3.8.4, building a schedule which takes into account the uncertainty of available bandwidth is expected to give robust schedules for workflow ensembles as well. Importantly, this procedure does not change the scheduling algorithm or the scheduler itself, it is simply a procedure that assists the scheduler in producing robust schedules under imprecise bandwidth estimations.

Computation of the Deflating $\mathcal{U}$ Value

Let $\mathcal{H}_{\mathcal{E},n}$ be a data set of previous executions of a specific workflow ensemble type $\mathcal{E}$, where each ensemble is composed of n inter-related DAGs. Each row of the data set $\mathcal{H}_{\mathcal{E},n}$ is represented by 7-tuples $\langle func, b, \mathcal{U}, p, m, c, j \rangle$, where the first three elements are information used to produce the ensemble's schedule, while the others are obtained from the ensemble's execution. The elements $func$, b and $\mathcal{U}$ are, respectively, the fitness function employed into the RobWE scheduler to produce the schedules, the measured available bandwidth at scheduling time, and the calculated $\mathcal{U}$ used to deflate the respective b value. The fourth element, p, represents the maximum percentage error between b and the observed available bandwidth during the execution of the ensemble $\mathcal{E}$. The observed overall makespan (execution time) of the ensemble is represented by the fifth element m, while the actual (real) cost of this execution is included as the sixth element c. The seventh and last element, j, corresponds to the Jain's fairness index of the ensemble execution, an index developed for resource sharing problems [33]. Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a set of numbers, the Jain's fairness index of $\mathcal{X}$ is defined as: $\frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n\left(\sum_{i=1}^{n} x_i^2\right)}$, where the index ranges between $\frac{1}{n}$ (worst case, unfair) and 1 (best case, fair).

The computation of the deflating $\mathcal{U}$-value involves the construction of a specific data subset $\mathcal{H}_{\mathcal{E},n}^{func} \subseteq \mathcal{H}_{\mathcal{E},n}$ for each fitness function $func$ in $\mathcal{H}_{\mathcal{E},n}$. This subset is composed of 3-tuples $\langle b, p, \mathcal{U} \rangle$ and is constructed as follows: for each pair (b, p) in $\mathcal{H}_{\mathcal{E},n}$, one 7-tuple from $\mathcal{H}_{\mathcal{E},n}$ is selected and its $\mathcal{U}$ value is taken to compose the 3-tuple $\langle b, p, \mathcal{U} \rangle$, which is stored in $\mathcal{H}_{\mathcal{E},n}^{func}$. The selection of this 7-tuple, however, is carried out according to the fitness function $func$ in $\mathcal{H}_{\mathcal{E},n}^{func}$:

- If $func$ is the fitness function f_1, the 7-tuple selected is the tuple that contains the smallest m value (the fastest execution), besides the b and p values
- If $func$ is the fitness function g_1, the 7-tuple selected is the tuple that contains the highest j value (the fairness execution), besides the b and p values
- If $func$ is the fitness function h_1, the 7-tuple selected is the tuple that contains the smallest c value (the cheapest execution), besides the b and p values

The functions f_1, g_1, and h_1 are presented in Table 4.2. The function g_2, which promotes the fairness maximisation, was not considered here because g_1 resulted in better results, as discussed in Section 4.6.2. Also, the motivation here is not to evaluate the RobWE's flexibility to allow the replacement of functions but to evaluate the RobWE's robustness under uncertainty of available bandwidth.

Each 3-tuple $\langle b, p, \mathcal{U} \rangle$ of the subset $\mathcal{H}_{\mathcal{E},n}^{func}$ means that, when the bandwidth estimate b had an expected uncertainty of up to p and the objective function used to produce the schedules was $func$, the scheduler was able to produce better results when the $\mathcal{U}$ value was used to deflate b. The calculation of the $\mathcal{U}$ value is carried out by a multiple linear regression (MLR) procedure, which uses the subset $\mathcal{H}_{\mathcal{E},n}^{func}$ as input data. The MLR uses the subset to compute the values of the coefficients a_1, a_2 and a_3 in the equation $f(x, y) = a_1 x + a_2 y + a_3$. In this equation, x is a variable that represents the currently

predicted uncertainty in the available bandwidth, y is an independent variable that represents the currently available bandwidth, and $\mathcal{U} = f(x, y)$. The subset construction and the computation of the coefficients a_1, a_2, and a_3 are illustrated in Algorithm 12.

Algorithm 12 Computation of the coefficients a_1, a_2, and a_3 for the equation $f(x, y)$

$\mathcal{H}_{\mathcal{E},n}$ = Database of ensembles $\mathcal{E}$ composed of n inter-related DAGs
f' = Fitness function employed in the scheduler
$\mathcal{H}_{\mathcal{E},n}^{f'} = \emptyset$ ▷ Subset for the fitness function f'
for all (b', p') pair existing in $\mathcal{H}_{\mathcal{E},n}$ **do**
    $ST = None$ ▷ the selected 7-tuple according to b', p', and f' values
    for all 7-tuples $\langle func, b, p, \mathcal{U}, m, c, j \rangle \in \mathcal{H}_{\mathcal{E},n}$ **do**
        if $f' == f_1$ **then**
            ST = Select the 7-tuple that $b = b'$, $p = p'$ and m is minimal
        else if $f' == g_1$ **then**
            ST = Select the 7-tuple that $b = b'$, $p = p'$ and j is maximal
        else if $f' == h_1$ **then**
            ST = Select the 7-tuple that $b = b'$, $p = p'$ and c is minimal
        end if
    end for
    $\mathcal{U}'$ = Take the $\mathcal{U}$ value from the selected 7-tuple represented by ST
    $\mathcal{H}_{\mathcal{E},n}^{f'} = \mathcal{H}_{\mathcal{E},n}^{f'} \cup \langle b', p', \mathcal{U}' \rangle$
end for
Call multiple linear regression by using the $\mathcal{H}_{\mathcal{E},n}^{f'}$ as an input data set and the function $f(x, y) = a_1 x + a_2 y + a_3$ as a formula to fit the data
Store the coefficient values a_1, a_2, and a_3

Applying the Procedure

When an ensemble $\mathcal{E}$ is about to be scheduled, the $\mathcal{U}$ value is computed through the p and b values obtained from the monitoring networking tools [14]. Making $x = p$, the currently predicted uncertainty, and $y = b$ the currently estimated available bandwidth, a percentage reduction factor $\mathcal{U} = f(x, y)$ is computed for the next ensemble schedule. The use of the procedure requires a set of historical data for training before the multiple linear regression approach can be applied.

4.7.2 Evaluation methodology

This section describes the methodology of the evaluation of the efficacy of the RobWE scheduler, a robust version of the original PSO-based scheduler for workflow ensembles under uncertainty of available bandwidth in links of a hybrid cloud. The purpose of this section is to compare the effectiveness of the RobWE scheduler with its original non-robust version that does not consider the presence of these uncertainties. Also, this evaluation further extends the evaluation of the proactivity of the procedure in being able to adequately deflate the unreliable bandwidth estimates, since we state that such estimates are provided by network tools that are not able to estimate the actual value of the available bandwidth. Therefore, this evaluation complements the evaluation of Sections 3.6 and 4.5.

Workflow Ensembles

Similar to Section 4.5.1, ensembles using workflows available from the workflow generator gallery[14] were employed. The gallery is composed of synthetic workflows modelled using patterns taken from real workflow applications [129] such as: LIGO, Montage, and SIPHT. As an initial evaluation, we considered only workflows with 50 tasks (DAG nodes). For each type of application, 100 different workflows were generated differing in the weights of the DAG nodes (computational demands) and DAG edges (communication demands). Using this collection of synthetic workflows, we constructed 500 ensembles for each workflow type, where each ensemble has 10 workflows.

Hybrid Cloud Scenario

The hybrid cloud scenario to execute the workflow ensembles is shown in Table 4.5, resulting in a total of 9 heterogeneous virtual machines (VMs). Inter-cloud bandwidths from 10 to 80 Mbps (in steps of 10 Mbps) and intra-cloud bandwidths of 100 Mbps were considered. We assume that the intra-cloud bandwidth is greater than the inter-cloud bandwidth, which is a reasonable assumption in real hybrid cloud environments. Lastly, as described in Section 4.3, we assume that VMs are exclusively used for the execution of workflow ensembles.

Table 4.5: Hybrid cloud scenario of the experimental evaluation of the RobWE scheduler.

Provider	Type	Cores	Performance per core	Price per unit of time	Number of VMs
A (public)	Small	2	2.0	$ 0.026	3
	Large	4	4.0	$ 0.120	1
B (public)	Large	2	4.0	$ 0.180	2
	X-Large	4	8.0	$ 0.300	2
C (private)	Small	2	1.0	$ 0.000	1

Experimental PSO Parameters

The JSwarm-PSO[15] is used to conduct the evaluation of the RobWE scheduler. The JSwarm-PSO configuration is similar to that described in Section 4.5.4. The number of particles was set to 500 and the number of iterations was set to 50. The particle's inertia ω was set to 1.5, and it decreases by 1% at each iteration. The particle's acceleration values a_1 and a_2 were set to 0.95, and both also decrease by 1% at each iteration.

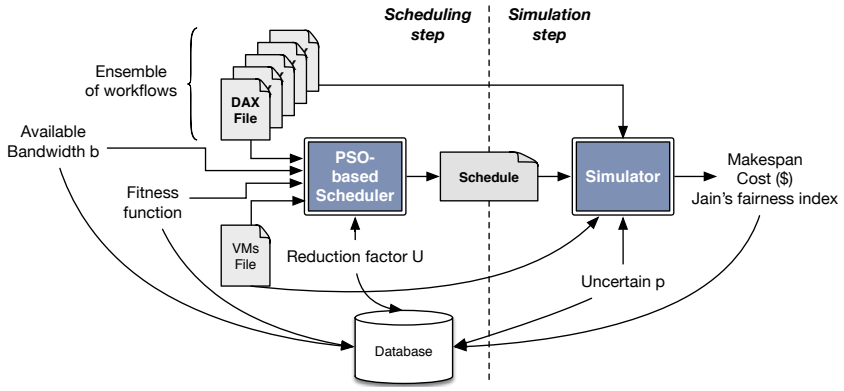
Simulator/Calculator

Similar to the evaluation of the original PSO-based scheduler for workflow ensemble, the data inputs for the RobWE scheduler are: DAX files (ensemble), a VMs file, and a fitness function (scheduler's objetive function). Each DAX file contains the description of the workflow in XML format, while the VMs file contains information about the hybrid resources from Table 4.5. After the schedule is produced by the scheduler, it is used as

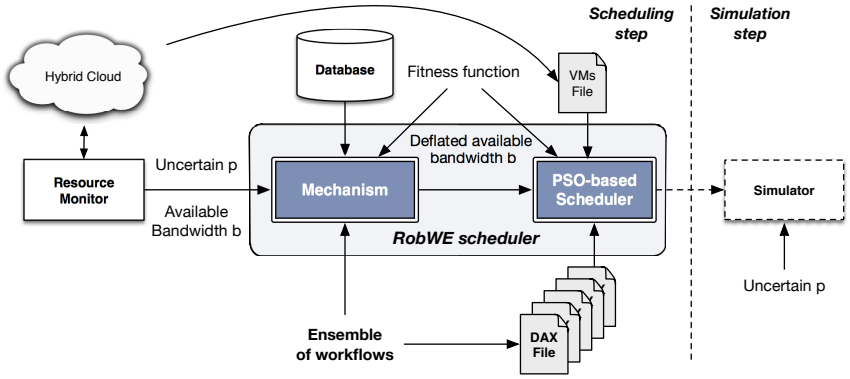
[14] `https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator`
[15] `http://jswarm-pso.sourceforge.net/`

an input to the simulator, which simulates the schedule and calculates the metric values, such as: the overall makespan (ensemble execution time), the slowdown values, and the monetary execution costs. However, at simulation time, all tasks of all workflows are re-ranked according to the execution finish time estimated by the produced schedule. For more details regarding the simulator, please see Appendix A.

The estimated inter-cloud available bandwidth value b is deflated by a factor $\mathcal{U}$, and the deflated value is used by the scheduler to produce the schedules; $\mathcal{U} = 0$ means that no reduction is applied, while $\mathcal{U} = 5$ means that the deflated bandwidth value provided to the scheduler is 95% of b. Importantly, at simulation time, no deflation is applied and the simulator uses a bandwidth estimate taken from the uniformly distributed interval $[b - p\% \,; b + p\%]$.



(a) First step of the experimental evaluation



(b) Second and third steps of the experimental evaluation

Figure 4.13: Overview of the steps of the RobWE scheduler experimental evaluation. These steps mirror the steps in Figure 3.10.

Experimental Setup

Following the evaluation of the procedure in Section 3.6, the RobWE scheduler is also evaluated in three steps. First (Fig. 4.13a), when the database is created, we assessed, without using the procedure, how fixed $\mathcal{U}$ factors ameliorate the negative impact of imprecise information about the inter-cloud available bandwidth on the execution of the workflow ensemble. The following fixed bandwidth deflating factors $\mathcal{U} \in \{0, 5, 15, 25, 35, 45, 50\}$

were used according to Section 3.6. For each value of $\mathcal{U}$, the percent error p ranged from 0% to 99%, in steps of 10%. Given a $\mathcal{U}$ value, a percentage error p is introduced by the simulator to derive the makespan, costs, and slowdowns of the ensemble execution, and all values are stored into the database. In this step, each application type is simulated using 500 different ensembles of 10 inter-related workflows each, 3 different fitness functions, 7 $\mathcal{U}$-values, 8 bandwidth values, and 11 different values of p. The database for each application type is produced from an extensive set of 924,000 simulations.

Secondly, the values of the the coefficients a_1, a_2 and a_3 in the equation $f(x, y) = a_1x + a_2y + a_3$ are computed using Algorithm 6. Lastly (Fig. 4.13b), to evaluate the effectiveness of the RobWE scheduler, new simulations were executed for each application type and fitness function by using its respective $\mathcal{U} = f(x, y)$. A total of 132,000 simulations were carried out in this step for each application type (500 different ensembles $\times$ 3 different fitness functions $\times$ 8 bandwidth values, and 11 p values).

4.7.3 Results

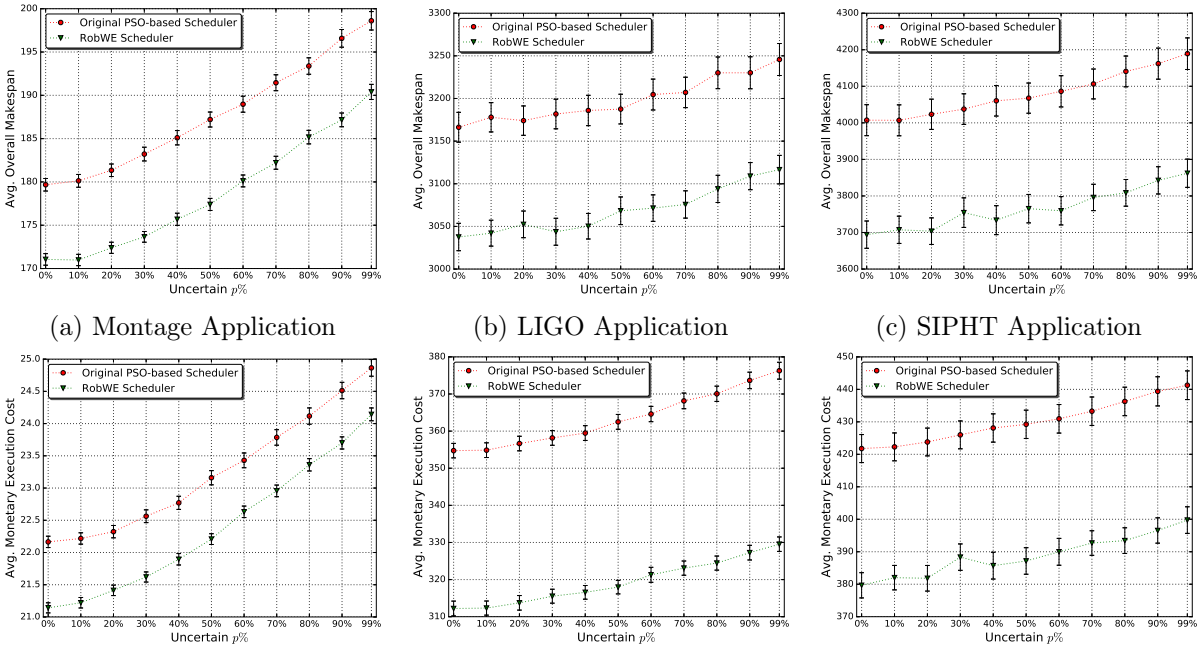
In this section, the effectiveness of the proposed RobWE scheduler is assessed and compared via simulation to that of its deterministic counterpart (the original PSO-based scheduler). This section focuses on the robustness of the RobWE scheduler for different degrees of uncertainties in available bandwidth estimates when scheduling workflow ensembles using several objective functions. The graphics shown are plotted over 500 simulations with a confidence interval of 95%.

RobWE as a Robust Makespan-aware Scheduler

Figures 4.14a, 4.14b, and 4.14c show the average overall makespan for the Montage, LIGO, and SIPHT applications, respectively, as a function of the degree of experienced uncertainty in the available bandwidth estimates. Each graph shows curves corresponding to the results given by the original PSO-based scheduler and the proposed RobWE scheduler. To comply with the objective of minimising the overall makespan, we employ the fitness function f_1 (Table 4.2) to drive PSO in both schedulers.

It can be seen in Figure 4.14 that, for all applications, the RobWE scheduler was able to produce a lower overall makespan than the original PSO-based scheduler regardless of the degree of uncertainty p. For example, when a measured available bandwidth of 30 Mbps was deflated by the $\mathcal{U}$ value with an uncertainty degree of 50%, the average values of the overall makespan produced by the proposed RobWE scheduler for the Montage, LIGO, and SIPHT applications were approximately 7%, 6%, and 10% lower than the values produced by the original PSO-based scheduler, respectively. Even in scenarios with high uncertainties, a remarkable difference between both schedulers was found for all applications. For example, for the SIPHT application, the RobWE scheduler reveals an average overall makespan 10% lower than those furnished by the original PSO-based scheduler when the uncertainty $p = 99\%$.

These results give some evidence of the robustness of the RobWE scheduler and its ability to achieve a lower makespan under uncertainties of the available bandwidth, since its performance underwent less degradation in makespan than the original PSO-based



(a) Montage Application (b) LIGO Application (c) SIPHT Application

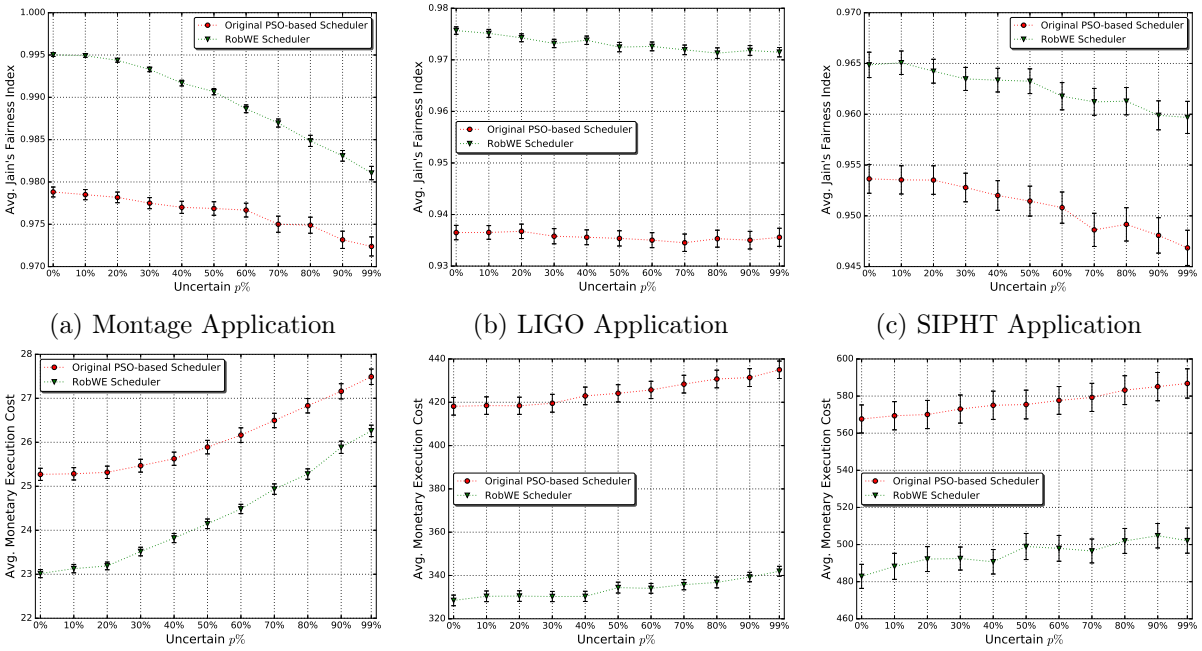
(d) Montage Application (e) LIGO Application (f) SIPHT Application

Figure 4.14: Average values of overall makespan (top) and execution cost (bottom) for 10-workflow ensembles for Montage, LIGO, and SIPHT applications when the measured available bandwidth in inter-cloud links was of 30 Mbps. These results were generated using the fitness function f_1 in both schedulers. For the proposed RobWE scheduler only, the measured available bandwidth of 30 Mbps was deflated by the procedure and provided as input to the scheduler. The x-axis represents the estimated error p in the measured available bandwidth which varies from 0% to 99%.

scheduler as the degree of uncertainty increases. As evidence that the RobWE scheduler did not result in more expensive execution, Figures 4.14d, 4.14e, and 4.14f show the average execution costs of the evaluation displayed in Figures 4.14a, 4.14b, and 4.14c, respectively. These figures also show that the performance of the proposed scheduler underwent less degradation in execution cost than that of the original PSO-based scheduler. For example, when $p = 50\%$, the RobWE scheduler was able to achieve schedules with reduced overall makespan for the Montage, LIGO, and SIPHT applications; it was also able to reduce costs in 5%, 13%, and 11%, respectively, from those produced by the original PSO-based scheduler. Indeed, the performance of the original scheduler degrades more than the RobWE scheduler as the degree of uncertainty increases.

RobWE as a Robust Fairness-aware Scheduler

This section presents the results obtained when using the fitness function g_1 (Table 4.2) in an attempt to maximize the fairness of the schedules. Figures 4.15a, 4.15b, and 4.15c show the average Jain's fairness index of the evaluation results for the Montage, LIGO, and SIPHT applications, respectively, as a function of the degree of uncertainty experienced in bandwidth estimates. For both schedulers, the function g_1 is able to drive PSO in achieving fairer schedules for all applications, since average values of Jain's fairness index



(a) Montage Application (b) LIGO Application (c) SIPHT Application

(d) Montage Application (e) LIGO Application (f) SIPHT Application

Figure 4.15: Average values of Jain's fairness index (top) and execution costs (bottom) for 10-workflow ensembles for the Montage, LIGO, and SIPHT applications when the measured available bandwidth in inter-cloud links was 30 Mbps. These results were generated using the fitness function h in both schedulers in an attempt to maximise fairness. For the proposed RobWE scheduler only, the measured available bandwidth of 30 Mbps was deflated by the procedure and provided as input to the scheduler. The x-axis represents the estimated error p in the measured available bandwidth which varies from 0% to 99%.

closer to 1 indicate that fair schedules were produced.

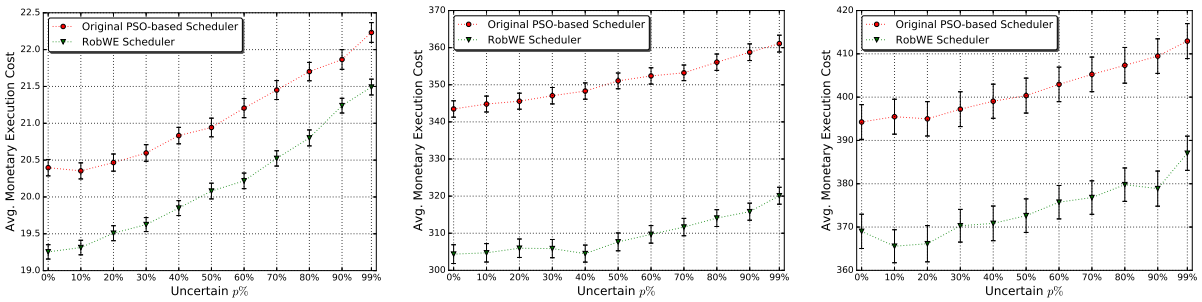
Similar to previous results, the performance of the RobWE scheduler had a smaller degradation in fairness than that of the original PSO-based scheduler. For instance, for an uncertainty degree of 20%, the average values of Jain's index produced by the RobWE scheduler for the Montage, LIGO, and SIPHT applications were approximately 3%, 5%, and 2% higher than the values produced by the original PSO-based scheduler, respectively. Although the evidence of smaller fairness degradation by the RobWE scheduler is modestly noticeable, the cost savings for these schedules are remarkably significant, as shown in Figures 4.15d, 4.15e, and 4.15f. For example, for $p = 20\%$, the results achieved by the RobWE scheduler were approximately 9%, 21%, and 16% cheaper than the results produced by the original PSO-based scheduler. The cost savings were noticeable because the robust schedules produced by the RobWE do not let the resources stay waiting for long file transmission that may be caused by unexpected high bandwidth variability.

A performance difference between both schedulers is also noticeable in cases with high uncertainties, wherein the RobWE scheduler underwent less degradation in its performance than the original PSO-based scheduler. For a projected uncertainty degree of 99%, for instance, the schedules achieved by the RobWE scheduler were approximately 2%, 4%, and 3% fairer for the Montage, LIGO, and SIPHT applications than the schedules produced by the original PSO-based scheduler, respectively. Regarding the execution costs,

these fairer schedules have offered considerable economy savings, approximately 6%, 10% and 11%, respectively. Again, the trend observed in Figure 4.14 can also be observed in Figure 4.15, suggesting that the original PSO-based scheduler underwent more degradation in fairness than the proposed RobWE scheduler in the presence of any degree of uncertainty p in the available bandwidth estimates.

RobWE as a Robust Cost-aware Scheduler

To comply with the objective function of minimising the execution cost, we employ the fitness function h_1 (Table 4.2) in the evaluation. Figures 4.16a, 4.16b, and 4.16c show the average monetary execution costs of the evaluation results for the Montage, LIGO, and SIPHT applications, respectively, as a function of the degree of uncertainty experienced in available bandwidth estimates. It is expected that the fitness function h_1 would drive the PSO algorithm in producing cheaper schedules than the previous fitness function f_1, and indeed, for all applications, this expectation has been achieved for both schedulers, as we can compare with the average costs displayed in Figures 4.14d, 4.14e, and 4.14f. Since the aim of this thesis is to discuss the negative consequences of imprecise information about the available bandwidth in the production of schedules and focus on how the RobWE scheduler can reduce such consequences, we do not go further into this analysis. For more details about the differences among fitness functions, please, refer to Section 4.4.4.



(a) Montage Application (b) LIGO Application (c) SIPHT Application

Figure 4.16: Average values of execution costs for 10-workflow ensembles for the Montage, LIGO, and SIPHT applications when the measured available bandwidth in inter-cloud links was of 30 Mbps. These results were generated using the fitness function h_1 in both schedulers. For the proposed RobWE scheduler only, the measured available bandwidth of 30 Mbps was deflated by the procedure and provided as input to the scheduler. The x-axis represents the estimated error p in the measured available bandwidth which varies from 0% to 99%.

When the fitness function h_1 is employed to minimise costs, the RobWE scheduler tends to produce cheaper schedules than the original PSO-based scheduler regardless of the degree of uncertainty p for all applications. For instance, for an uncertainty degree of 30%, the average execution cost produced by the RobWE scheduler for the Montage, LIGO, and SIPHT applications was approximately 6%, 14%, and 10% lower than that produced by the original PSO-based scheduler, respectively. A remarkable difference between both schedulers is also observed in scenarios with high uncertainties. For instance,

when $p = 99\%$, the results achieved by the RobWE scheduler were 5%, 12%, and 6% cheaper than the results produced by the original PSO-based scheduler.

Similar to the previous analyses, the results presented in this section also reinforce the argument about robustness of the RobWE scheduler in achieving cheaper executions in the presence of uncertainties in the available bandwidth, since its performance also underwent less degradation in cost than the performance of the original PSO-based scheduler as the degree of uncertainty increases. The trend observed in Figure 4.14, which can also be observed in Figures 4.16 and 4.15, shows a significant degradation in performance for the original PSO-based scheduler than the performance of the RobWE scheduler as the degree of uncertainty increases.

4.7.4 Discussion

This section introduced the RobWE scheduler, a scheduler that extends the robustness of the original PSO-based scheduler for workflow ensembles presented in Section 4.3 with the employment of the procedure proposed in Chapter 3, which attempts to improve the scheduler's robustness under uncertainties of available bandwidth in links of a hybrid cloud. As it was expected, poor schedules were produced by the original PSO-based scheduler since it does not consider the variabilities of the available bandwidth that may occur during the execution of the workflow ensemble. As it was also expected, the procedure was able to considerably improve the flexible scheduler's robustness in the presence of such variabilities.

The difference in performance between the use and non-use of the "robustifier" procedure lies in the fact that, to produce robust schedules, the RobWE scheduler uses a deflated value of the measured available bandwidth according to an expected uncertainty of such measurement, while the original PSO-based scheduler uses the measured available bandwidth as a precise information. The results have indicated that the effectiveness of the RobWE scheduler relies on its ability to cope with a high level of uncertainty. The performance of the original PSO-based scheduler fed by imprecise information underwent significant degradation than the performance of RobWE for all evaluated fitness functions. As workflow applications, especially those grouped into ensembles, generate huge amounts of data during their executions, the scheduler proposed in this thesis seems to be attractive for hybrid cloud environments, where the available bandwidth in inter-cloud links may vary and cannot be estimated precisely in advance. Results of simulations considering diverse scenarios, based on several degrees of uncertainty in available bandwidth estimates, characteristics of bandwidth estimations and workflow applications, demonstrate the advantages of the RobWE scheduler.

The results obtained in this section reinforces those results presented in Sections 3.7 and 3.8.4, where the procedure has been able to considerably improve the effectiveness of deterministic single-workflow schedulers under imprecise bandwidth estimates, since they have started producing robust schedules that underwent remarkable less degradation in performance than that of given by the same schedulers without using our procedure. The results shown in this section indicate that the employment of the procedure was able to considerably reduce the impact of using misleading bandwidth estimates in the production

of schedules for workflow ensembles.

4.8 Final Remarks

A scheduler is considered *flexible* when it allows the replacement of the objective function according to the user's needs. In this case, the scheduler's objective function can be seen as a "module" that can be switched when necessary, making the proposed scheduler usable in various scenarios, such as saving cost expenses with executions of applications using pay-as-you-go cloud resources or sharing the resources equally among the workflows of an ensemble. To the best of our knowledge, there is no work in the literature that proposes a workflow scheduler that promotes this flexibility feature, especially a scheduler for workflow ensembles using cloud computing resources.

This chapter proposed a flexible scheduler to schedule a set of workflows grouped into ensembles. To produce the schedules, the proposed scheduler employs a method called Particle Swarm Optimisation (PSO), a population-based heuristic global-search algorithm. This scheduler is accomplished by modelling the workflow scheduling problem as a PSO and developing a fitness function that acts as the scheduler's objective function. Due to the possibility to create as many fitness functions as desired, this scheduler is considered flexible since it allows the replacement of the objective function according to the user's needs, making the scheduler flexible enough to be usable in various several scenarios, from minimising the application's execution time to minimising the application's execution cost. In this case, minimising time and cost are two examples of conflicting objective functions that, in general, the user must use a specialised (and probably different) scheduler to meet each goal for running the application. The proposed scheduler, however, can comply with both objectives due to the proposed flexibility feature, making it easier for the user to use only one scheduler to meet several conflicting objective functions.

Given a set of scientific workflow ensembles, the proposed flexible scheduler was firstly evaluated using three types of objective functions to produce the schedules: minimisation of the overall makespan, maximisation of fairness, and minimisation of costs. Based on these objectives, four fitness functions were developed and employed: the function f_1 aims to lead the PSO in achieving solution that minimises the overall makespan, while the functions g_1 and g_2 aim to maximise the fairness among the workflows within a set. Lastly, the function h_1 attempts to lead the PSO in producing low-cost schedules. The proposed scheduler was evaluated via simulation using ensembles of synthetic workflows, which were generated based on statistics from real scientific applications. The results of this first experimental evaluation indicate that the desired flexibility for the proposed PSO-based scheduler has been achieved since each fitness function could produce schedules that satisfied its objective. The function f_1 was the function that was able to obtain schedules with smallest overall makespan, while g_1 and g_2 were the functions that were able to achieve the fairest schedules when compared to the schedules produced by f_1 and h_1. Finally. the function h_1 was the only one that could achieve the cheapest schedules. Indeed, the proposed flexible scheduler provides a freedom to the user to be able to design different objective functions to manage the production of schedules of their applications

in different ways and according to their needs. The flexible scheduler promotes, let's say, the $n : 1$ *model*, where the user can employ n different objectives using only 1 scheduler.

The performance of workflow execution in hybrid clouds depends on the available bandwidth in the inter-cloud links since this bandwidth determines the data transfer time between tasks residing in different clouds. The available bandwidth in these links can fluctuate considerably since Internet links are shared resources. This fluctuation contributes to imprecision estimates of the available bandwidth, and furnishing such inaccurate estimates as an input to a scheduler usually results in the production of poor schedules. The proposed scheduler is flexible, however, it does not include a procedure to produce robust schedules under uncertainties of available bandwidth. For this purpose, it was employed in Section 4.7 the procedure presented in Chapter 3. In this second evaluation, the negative impact of uncertainties on the original PSO-based scheduler was used to justify the inclusion of a proactive approach (procedure) into the PSO-based scheduler (called as RobWE scheduler) to schedule workflow ensembles involving uncertainty in inter-cloud available bandwidth. The results show that the performance of the RobWE scheduler undergoes less degradation than that of the original PSO-based scheduler as the degree of uncertainty increases.

This study suggests several studies for future work. Our current experimental evaluation employed only ensembles composed of workflows in which all of them have the same number of tasks, which in this case was 50 tasks. In the future, we plan to extend this experimental evaluation by using different ensemble types, such as ensembles that contain a small number of larger workflows (1000 tasks, for instance) and a large number of smaller workflows (50 tasks for example). In other words, ensembles to have a slightly larger number of large workflows, which reflects behaviour commonly observed in many computational scientific applications' workloads. Furthermore, in a dynamic approach when it is necessary to switch the scheduling context on-the-fly, for example, the replacement of the objective function could occur during the scheduling of the ensemble. A study involving a dynamic approach for the proposed scheduler is also left as future work. Regarding the RobWE, although the procedure introduced was a preventive approach for the handling of imprecise information, it was not designed to replace reactive approaches, such as self-adjusting scheduling. The integration of the two approaches seems to be a promising option and is left as future work as well.

Chapter 5

Conclusion

A computational environment in which bandwidth estimates are very likely to vary and can not be estimated accurately in advance by networking tools makes it difficult to provide accurate information about the bandwidth available to workflow schedulers. This thesis presented a procedure that guarantees a good functioning of existing workflow schedulers in the presence of inaccuracies in input data, especially information regarding bandwidth estimates furnished by these tools. Indeed, these schedulers are expected to malfunction when using imprecise input data since they were not designed to produce good schedules using inaccurate information, as well as they do not have any type of embedded procedure to address this uncertainty issues.

Chapter 2 introduced the background and key concepts needed to understand the content of this thesis. The definition of the cloud computing was presented, especially the description of hybrid clouds used in this thesis. Scientific applications modelled as workflows of tasks and represented by direct acyclic graphs (DAGs) were also described, including their characteristic of containing loosely-coupled computational tasks stitched by data- and control-flow dependencies through files. Chapter 2 also presented the general definition of the workflow scheduling problem, which is known to be NP-Complete, and highlighted the main characteristics of deterministic workflow schedulers that assume that all input data is known with accuracy in advance. As it was discussed, in many practical scheduling situations, the input data provided to the scheduler is likely to be uncertain, especially those data regarding available bandwidth in channels that connect clouds. Due to this, Chapter 2 finalised it by posing the importance of considering the accuracy of input data to deterministic schedulers. The precision of the input data that are provided to the scheduler is essential to the effectiveness of application execution performance when using the produced schedule. This discussion served as a motivation to the development a procedure to mitigate the production of poor schedules caused by the use of imprecise bandwidth estimates.

Since deterministic schedulers were not designed to address inaccuracies in the input data, it makes them prone to produce schedules that are sensitive (non-robust) to these uncertainties. On a contractionary basis, these schedulers are specially designed to work in an environment where the available bandwidth is very likely to vary and cannot be estimated accurately in advance by today's monitoring networking tools. Chapter 3 highlighted that the uncertainties of available bandwidth can be a result of imprecise

measurement and monitoring network tools and/or their incapacity of estimating the real value of the available bandwidth during the workflow scheduling time. Actually, the bandwidth estimation is given in ranges rather than deterministic values, and therefore the schedules are produced based on imprecise bandwidth estimates. This makes a deterministic scheduler to produce poor schedules that can result in ineffective workflow execution performance. Chapter 3 presented a procedure to provide robustness for non-robust schedulers be able to produce good schedules under inaccurate bandwidth estimates. The procedure was evaluated using the HCOC [35] and adapted-PSO-based [104] deterministic schedulers presented in the literature. Both schedulers are classified as a deadline- and cost-aware scheduler, and therefore they focus on minimising the monetary execution costs as well as in satisfying specified deadlines for such executions. According to the simulations, the employment of the procedure on both schedulers was able to reduce the number of disqualified solutions (solutions that missed the deadline) by up to 70% with costs reduced by up to 10%. The procedure has been able to considerably improve the effectiveness of both schedulers under imprecise bandwidth estimates since they have started producing robust schedules that underwent remarkable less degradation in performance than that of given by the same schedulers without using our procedure. Therefore, experiments performed with these two schedulers using several workflows evidenced the effectiveness of the procedure in providing a certain level of robustness ("robustifying") these schedulers in situations where the degree of uncertainty (variability) of bandwidth estimates is very high.

Chapter 3 also presented the experiments of the procedure using the IPDT [19], a deterministic non-robust makespan-aware scheduler. The key difference in this experiment from the previous one is that a robust version of the IPDT under uncertainty of bandwidth availability, called IP-FULL-FUZZY [19], was introduced in the evaluation for comparison purposes. The design of the IP-FULL-FUZZY was based on the IPDT but it employs fuzzy optimisation for the consideration of the uncertainties in the input data. The comparison is carried out using the *speedup*, a metric that correlates the ratio between the time taken to process sequentially the tasks of an application and the time to process the same set of tasks in a non-sequential fashion. The speedup values given by the IPDT-Mechanism (IPDT using the procedure) was compared and contrasted with the IP-FULL-FUZZY scheduler (without using the procedure). According to the results provided from simulations, the speedup produced by the IPDT-Mechanism is on average 25% higher than that of the IP-FULL-FUZZY, evidencing the effectiveness of the proposed procedure in enhancing the robustness of the IPDT better than that of the IP-FULL-FUZZY, which employs fuzzy optimisation for this purpose. In other words, the performance of the IP-FULL- FUZZY fed by imprecise bandwidth estimates underwent significant speedup degradation than did the one of the IPDT-Mechanism, evidencing again the effectiveness of the procedure in "robustifying" non-robust schedulers.

Chapter 4 presented a flexible scheduler for workflows in an ensemble, a type of scientific applications that comprises a set of inter-related workflows instead of being represented by a single workflow. The flexibility of the scheduler comes from the possibility of replacing the objective function according to the user's needs, making the scheduler usable in various scenarios, such as saving cost expenses with an ensemble execution or

sharing the resources equally among the workflows in an ensemble. The flexibility is given by the possibility of modelling the scheduler as a particle swarm optimisation (PSO) and its objective function as a PSO's fitness function. Three types of objective function were evaluated: minimisation of the overall makespan, maximisation of fairness, and minimisation of costs. Based on these objectives, four fitness functions were developed and employed on the PSO. The function f_1 aims to lead the PSO in achieving solution that minimises the overall makespan of the ensemble, while the functions g_1 and g_2 aim to maximise the fairness among the workflows. Lastly, the function h_1 attempts to lead the PSO in producing low-cost schedules. The results of our experimental evaluation indicate that the desired flexibility for the scheduler has been achieved since each fitness function could produce schedules that satisfied its objective.

Although the novelty of the flexible scheduler is the replacement of objective functions according to the user's needs, it is still a non-robust scheduler under inaccuracies in input data. To generate robust schedules under imprecise bandwidth estimations, Chapter 4 applied the procedure of Chapter 3 into the original PSO-based flexible scheduler and creates the RobWE, a robust scheduler for workflow ensembles. The results show that the performance of the RobWE scheduler undergoes less degradation than that of the original PSO-based scheduler as the degree of uncertainty increases across all objective functions evaluated. Again, the effectiveness of the procedure in furnishing a certain level of robustness for schedules produced by non-robust schedulers under imprecise bandwidth estimates was experimentally evidenced. As workflow applications, especially those grouped into ensembles, generate huge amounts of data during their executions, the RobWE scheduler seems to be attractive for hybrid cloud environments, where the available bandwidth in communication links connecting resources in different sites may vary and cannot be estimated precisely in advance.

The scheduling of workflows in shared and heterogeneous systems is a theme that can still be approached in different ways, which opens space for a diversity for a new research horizon. This thesis suggests several studies for future work. The following list presents some points related to the content of this thesis that merit future research:

- The effectiveness and efficiency of the procedure in providing robustness to non-robust schedulers are evaluated through simulation. However, it should be interesting to run experiments on a real testbed of a hybrid cloud environment to prove the effectiveness of the procedure in providing robustness for non-robust schedulers, in addition to bringing improvements and adjustments to the procedure itself;

- We also intend to improve the performance of the proposed procedure by updating the historical information as well as by deriving criteria for selecting obsolete information to be deleted from the database in order to maintain the scalability of the procedure usability;

- The procedure introduced here represents a preventive approach for the handling of imprecise information, but it was not designed to replace reactive approaches, such as self-adjusting schemes. The integration of the two approaches seems to be a promising topic for a future work;

- Regarding the flexible scheduler for workflow ensembles in a dynamic approach, when it is necessary to switch the scheduling context on-the-fly, for example, the replacement of the objective function could occur during the scheduling of the ensemble. A study involving a dynamic approach for the proposed scheduler is also left as future work;

- The scheduling of workflow ensembles in a dynamic approach using a procedure that employs the integration of preventive and reactive approaches is another interesting topic to produce a robust scheduler;

- Researches that includes multiple workflow scheduling is undoubtedly necessary. An evaluation of the flexible scheduler involving a set of non inter-related workflows, i.e. multiple workflows, is left as future work too;

Bibliography

[1] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema. Cost-Driven Scheduling of Grid Workflows Using Partial Critical Paths. *IEEE Transactions on Parallel and Distributed Systems*, 23(8):1400–1414, Aug 2012.

[2] Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. Survey Cloud Monitoring: A Survey. *Comput. Netw.*, 57(9):2093–2115, June 2013.

[3] S. Akioka and Y. Muraoka. HPC Benchmarks on Amazon EC2. In *IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, pages 1029–1034, April 2010.

[4] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A Scalable, Commodity Data Center Network Architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.

[5] Gabrielle Allen, David Angulo, Ian Foster, Gerd Lanfermann, Chuang Liu, Thomas Radke, Ed Seidel, and John Shalf. The Cactus Worm: Experiments with Dynamic Resource Discovery and Allocation in a Grid Environment. *The International Journal of High Performance Computing Applications*, 15(4):345–358, 2001.

[6] Demetres Antoniades, Manos Athanatos, Antonis Papadogiannakis, Evangelos P. Markatos, and Constantine Dovrolis. Available bandwidth measurement as simple as running wget. In *Passive and Active Measurement (PAM) Conference*, page 61–70., 2006.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Journal of Communication ACM*, 53:50–58, apr 2010.

[8] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Commun. ACM*, 53(4):50–58, April 2010.

[9] T. Arsan. Review of bandwidth estimation tools and application to bandwidth adaptive video streaming. In *High Capacity Optical Networks and Emerging/Enabling Technologies*, pages 152–156, Dec 2012.

[10] Marcos Dias Assunção, Alexandre Costanzo, and Rajkumar Buyya. A cost-benefit analysis of using cloud computing to extend the capacity of clusters. *Cluster Computing*, 13:335–347, September 2010.

[11] A. O. Ayodele, J. Rao, and T. E. Boult. Towards Application-centric Fairness in Multi-tenant Clouds with Adaptive CPU Sharing Model. In *9th International Conference on Cloud Computing (CLOUD)*, pages 367–375, June 2016.

[12] Roger Barga and Dennis Gannon. *Scientific versus Business Workflows*, pages 9–16. Springer London, London, 2007.

[13] D. M. Batista, C. G. Chaves, and N. L. S. da Fonseca. Embedding Software Requirements in Grid Scheduling. In *IEEE International Conference on Communications (ICC)*, pages 1–6, June 2011.

[14] Daniel M. Batista, Luciano J. Chaves, Nelson L. Fonseca, and Artur Ziviani. Performance Analysis of Available Bandwidth Estimation Tools for Grid Networks. *The Journal of Supercomputing*, 53(1):103–121, July 2010.

[15] Daniel M. Batista, Nelson L. S. da Fonseca, and Flavio K. Miyazawa. A Set of Schedulers for Grid Networks. In *ACM Symposium on Applied Computing (SAC)*, pages 209–213, 2007.

[16] Daniel M. Batista and Nelson L.S. da Fonseca. Robust scheduler for grid networks under uncertainties of both application demands and resource availability. *Computer Networks*, 55(1):3 – 19, 2011.

[17] Daniel M. Batista, Nelson L.S. da Fonseca, Flavio K. Miyazawa, and Fabrizio Granelli. Self-adjustment of resource allocation for grid applications. *Computer Networks*, 52(9):1762 – 1781, 2008.

[18] Daniel M. Batista, André C. Drummond, and Nelson L. S. da Fonseca. Scheduling Grid Tasks Under Uncertain Demands. In *ACM Symposium on Applied Computing (SAC)*, pages 2041–2045, 2008.

[19] Daniel Macedo Batista. *Escalonadores de Tarefas Dependentes para Grades Robustos às Incertezas das Informações de Entrada*. PhD thesis, Instituto de Computação (IC) da Universidade Estadual de Campinas (Unicamp), Campinas, SP, Brasil, Fevereiro 2010.

[20] G. Bruce Berriman, Ewa Deelman, Gideon Juve, Mats Rynge, and Jens-S. Vöckler. The application of cloud computing to scientific workflows: a study of cost and performance. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983), 2013.

[21] K. Bessai, S. Youcef, A. Oulamara, Claude Godart, and S. Nurcan. Bi-criteria Workflow Tasks Allocation and Scheduling in Cloud Computing Environments. In *IEEE 5th International Conference on Cloud Computing (CLOUD)*, pages 638–645, 2012.

[22] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. H. Su, and K. Vahi. Characterization of scientific workflows. In *3rd Workshop on Workflows in Support of Large-Scale Science*, pages 1–10, Nov 2008.

[23] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana. Towards Virtual Machine Migration in Fog Computing. In *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, pages 1–8, Nov 2015.

[24] L.F. Bittencourt, E.R.M. Madeira, and N.L.S. Da Fonseca. Impact of communication uncertainties on workflow scheduling in hybrid clouds. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1623–1628, Dec 2012.

[25] L.F. Bittencourt, R. Sakellariou, and E. R M Madeira. DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm. In *8th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 27–34, 2010.

[26] L.F. Bittencourt, C.R. Senna, and E.R.M. Madeira. Enabling execution of service workflows in grid/cloud hybrid systems. In *IEEE/IFIP Network Operations and Management Symposium Workshops*, pages 343 –349, april 2010.

[27] L.F. Bittencourt, C.R. Senna, and E.R.M. Madeira. Scheduling service workflows for cost optimization in hybrid clouds. In *International Conference on Network and Service Management (CNSM)*, pages 394 –397, oct. 2010.

[28] Luiz F. Bittencourt and Edmundo R. M. Madeira. Fulfilling Task Dependence Gaps for Workflow Scheduling on Grids. In *3rd International IEEE Conference on Signal-Image Technologies and Internet-Based System*, pages 468–475, Washington, DC, USA, 2007. IEEE Computer Society.

[29] Luiz F. Bittencourt and Edmundo R. M. Madeira. A Performance-oriented Adaptive Scheduler for Dependent Tasks on Grids. *Journal of Concurrency and Computation: Practice & Experience - Middleware for Grid Computing: Future Trends*, 20(9):1029–1049, June 2008.

[30] Luiz F. Bittencourt, Edmundo R. M. Madeira, and Nelson L. S. da Fonseca. *Resource Management and Scheduling*, pages 243–267. John Wiley & Sons, Inc, 2015.

[31] Luiz F. Bittencourt, Rizos Sakellariou, and Edmundo R. M. Madeira. Using Relative Costs in Workflow Scheduling to Cope with Input Data Uncertainty. In *10th International Workshop on Middleware for Grids, Clouds and e-Science (MGC)*, pages 8:1–8:6, 2012.

[32] Luiz Fernando Bittencourt. *Algoritmos para Escalonamento de Tarefas Representadas por Grafos Acíclicos Direcionados em Grades Computacionais*. PhD thesis, Instituto de Computação (IC) da Universidade Estadual de Campinas (Unicamp), Campinas, SP, Brasil, Fevereiro 2010.

[33] Luiz Fernando Bittencourt and Edmundo R. M. Madeira. Towards the Scheduling of Multiple Workflows on Computational Grids. *Journal of Grid Computing*, 8(3):419–441, 2010.

[34] Luiz Fernando Bittencourt, Edmundo R. M. Madeira, and Nelson L. S. da Fonseca. Scheduling in hybrid clouds. *IEEE Communications Magazine*, 50(9):42–47, 2012.

[35] Luiz Fernando Bittencourt and Edmundo Roberto Mauro Madeira. HCOC: A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds. *Journal of Internet Services and Applications*, 2(3):207–227, 2011.

[36] Klavdiya Bochenina, Nikolay Butakov, and Alexander Boukhanovsky. Static Scheduling of Multiple Workflows with Soft Deadlines in Non-dedicated Heterogeneous Environments. *Future Gener. Comput. Syst.*, 55(C):51–61, February 2016.

[37] M. R. Bonyadi and Z. Michalewicz. Particle Swarm Optimization for Single Objective Continuous Space Problems: A Review. *Evolutionary Computation*, 25(1):1–54, March 2017.

[38] A. Botta, A. Pescape, and G. Ventre. On the performance of bandwidth estimation tools. In *Proceedings of Systems Communications*, pages 287–292, Aug 2005.

[39] Gerard Briscoe and Alexandros Marinos. Digital Ecosystems in the Clouds: Towards Community Cloud Computing. *CoRR*, abs/0903.0694, 2009.

[40] James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions. *Journal of Grid Computing*, 6(3):255–276, Sep 2008.

[41] Piotr Bryk, Maciej Malawski, Gideon Juve, and Ewa Deelman. Storage-aware Algorithms for Scheduling of Workflow Ensembles in Clouds. *Journal of Grid Computing*, 14(2):359–378, 2015.

[42] Scott Callaghan, Philip Maechling, Ewa Deelman, Karan Vahi, Gaurang Mehta, Gideon Juve, Kevin Milner, Robert Graves, Edward Field, David Okaya, Dan Gunter, Keith Beattie, and Thomas Jordan. Reducing Time-to-Solution Using Distributed High-Throughput Mega-Workflows - Experiences from SCEC CyberShake. In *4th IEEE International Conference on e-Science (e-Science 2008)*, 2008.

[43] Scott Callaghan, Philip Maechling, Patrick Small, Kevin Milner, Gideon Juve, Thomas H. Jordan, Ewa Deelman, Gaurang Mehta, Karan Vahi, Dan Gunter, and Keith Beattie. Metrics for Heterogeneous Scientific Workflows: A Case Study of an Earthquake Science Application. *International Journal of High Performance Computing Applications*, 25(3):274–285, August 2011.

[44] H. Casanova, A Legrand, D. Zagorodnov, and F. Berman. Heuristics for scheduling parameter sweep applications in grid environments. In *9th Heterogeneous Computing Wksh.*, pages 349–363, 2000.

[45] T. L. Casavant and J. G. Kuhl. A Taxonomy of Scheduling in General-purpose Distributed Computing Systems. *IEEE Trans. Softw. Eng.*, 14(2):141–154, February 1988.

[46] C. G. Chaves, D. M. Batista, and N. L. S. da Fonseca. Scheduling cloud applications under uncertain available bandwidth. In *IEEE International Conference on Communications (ICC)*, pages 3781–3786, June 2013.

[47] W. Chen, A. Fekete, and Y. C. Lee. Exploiting deadline flexibility in Grid workflow rescheduling. In *11th IEEE/ACM International Conference on Grid Computing*, pages 105–112, Oct 2010.

[48] W. Chen, R. F. D. Silva, E. Deelman, and R. Sakellariou. Balanced Task Clustering in Scientific Workflows. In *IEEE 9th International Conference on e-Science*, pages 188–195, Oct 2013.

[49] Wei Chen, Young Choon Lee, Alan Fekete, and Albert Y. Zomaya. Adaptive Multiple-workflow Scheduling with Task Rearrangement. *J. Supercomput.*, 71(4):1297–1317, Apr 2015.

[50] Douglas E. Comer. *Computer Networks and Internets*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2008.

[51] Rafael Ferreira da Silva, Weiwei Chen, Gideon Juve, Karan Vahi, and Ewa Deelman. Community Resources for Enabling Research in Distributed Scientific Workflows. In *IEEE International Conference on e-Science*, Nov 2014.

[52] Ewa Deelman, Scott Callaghan, Edward Field, Hunter Francoeur, Robert Graves, Nitin Gupta, Vipin Gupta, Thomas H. Jordan, Carl Kesselman, Philip Maechling, John Mehringer, Gaurang Mehta, David Okaya, Karan Vahi, and Li Zhao. Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance Tracking: The CyberShake Example. In *2nd IEEE International Conference on e-Science and Grid Computing (E-SCIENCE)*, pages 14–, Washington, DC, USA, 2006.

[53] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. Workflows and e-Science: An Overview of Workflow System Features and Capabilities. *Future Generation Computer Systems*, July 2008.

[54] Ewa Deelman, Gideon Juve, and G. Bruce Berriman. Using Clouds for Science, is it just Kicking the Can down the Road? In *2nd International Conference on Cloud Computing and Services Science (CLOSER)*, pages 127–134, 2012.

[55] Ewa Deelman, Carl Kesselman, Gaurang Mehta, Leila Meshkat, Laura Pearlman, Kent Blackburn, Phil Ehrens, Albert Lazzarini, Roy Williams, and Scott Koranda. GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists. In *High Performance Distributed Computing (HPDC)*, 2002.

[56] Ewa Deelman, Tom Peterka, Ilkay Altintas, Christopher D Carothers, Kerstin Kleese van Dam, Kenneth Moreland, Manish Parashar, Lavanya Ramakrishnan, Michela Taufer, and Jeffrey Vetter. The future of scientific workflows. *The International Journal of High Performance Computing Applications*, accepted, 2017.

[57] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good. The cost of doing science on the cloud: The montage example. In *ACM/IEEE Conference on Supercomputing*, pages 50:1–50:12, 2008.

[58] Ewa Deelman, Karan Vahi, Mats Rynge, Gideon Juve, Rajiv Mayani, and Rafael Ferreira da Silva. Pegasus in the Cloud: Science Automation through Workflow Technologies. *IEEE Internet Computing*, 20(1):70–76, 2016.

[59] Jack B. Dennis. Principles to support modular software construction. *Journal of Computer Science and Technology*, 32(1):3–10, Jan 2017.

[60] R. Duan, R. Prodan, and X. Li. Multi-Objective Game Theoretic Scheduling of Bag-of-Tasks Workflows on Hybrid Clouds. *IEEE Transactions on Cloud Computing*, 2(1):29–42, Jan 2014.

[61] Juan J. Durillo and Radu Prodan. Multi-objective workflow scheduling in Amazon EC2. *Cluster Computing*, 17(2):169–189, Jun 2014.

[62] Hamid Mohammadi Fard, Radu Prodan, Juan Jose Durillo Barrionuevo, and Thomas Fahringer. A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments. In *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2012)*, pages 300–309, Washington, DC, USA, 2012.

[63] Rafael Ferreira da Silva, Rosa Filgueira, Ilia Pietri, Ming Jiang, Rizos Sakellariou, and Ewa Deelman. A Characterization of Workflow Management Systems for Extreme-Scale Applications. *Future Generation Computer Systems*, 75:228–238, 2017.

[64] N. Fujimoto and K. Hagihara. Near-optimal dynamic task scheduling of precedence constrained coarse-grained tasks onto a computational grid. In *2nd International Symposium on Parallel and Distributed Computing.*, pages 80–87, Oct 2003.

[65] Dada Emmanuel Gbenga and Effirul Ikhwan Ramlan. Understanding the Limitations of Particle Swarm Algorithm for Dynamic Optimization Tasks: A Survey Towards the Singularity of PSO for Swarm Robotic Applications. *ACM Comput. Surv.*, 49(1):8:1–8:25, July 2016.

[66] Thiago A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira. Escalonamento de workflows com uso intensivo de dados em nuvens. In *XI Workshop de Computação em Clouds e Aplicações (WCGA) in conjunction with the Brazilian Symposium on Computer Networks and Distributed Systems (SBRC)*, Brasília, DF, Brasil, maio 2013.

[67] Thiago A. L. Genez, Luiz F. Bittencourt, Nelson L. S. Da Fonseca, and Edmundo R. M. Madeira. Refining the Estimation of the Available Bandwidth in Inter-Cloud Links for Task Scheduling. In *IEEE Global Communications Conference (GLOBECOM)*, Dec 2014.

[68] Thiago A. L. Genez, Luiz F. Bittencourt, Nelson L. S. da Fonseca, and Edmundo R. M. Madeira. Estimation of the Available Bandwidth in Inter-Cloud Links for Task Scheduling in Hybrid Clouds. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2015.

[69] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. Workflow Scheduling for SaaS/PaaS Cloud Providers Considering Two SLA Levels. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 906 –912, 2012.

[70] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. On the Performance-Cost Tradeoff for Workflow Scheduling in Hybrid Clouds. In *IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 411–416, Dec 2013.

[71] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. Using Time Discretization to Schedule Scientific Workflows in Multiple Cloud Providers. In *IEEE 6th International Conference on Cloud Computing*, pages 123–130, 2013.

[72] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. Time-discretization for Speeding-up Scheduling of Deadline-constrained Workflows in Clouds. *Journal of Future Generation Computer Systems*, Online, Elsevier Science Publishers B. V., 2017.

[73] Thiago A. L. Genez, Luiz F. Bittencourt, Rizos Sakellariou, and Edmundo R. M. Madeira. A Flexible Scheduler for Workflow Ensembles. In *9th IEEE/ACM International Conference on Utility and Cloud Computing*, pages 55–62, 2016.

[74] Thiago A. L. Genez, Luiz F. Bittencourt, Rizos Sakellariou, and Edmundo R. M. Madeira. A Robust Scheduler for Workflow Ensembles under Uncertainties of Available Bandwidth. In *10th IEEE International Conference on Cloud Computing*, 2017.

[75] Thiago A. L. Genez, I. Pietri, R. Sakellariou, L. F. Bittencourt, and E. R. M. Madeira. A Particle Swarm Optimization Approach for Workflow Scheduling on Cloud Resources Priced by CPU Frequency. In *IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 237–241, Dec 2015.

[76] Emanuele Goldoni and Marco Schivi. End-to-end Available Bandwidth Estimation Tools, an Experimental Comparison. In *2nd International Conference on Traffic Monitoring and Analysis*, pages 171–182, 2010.

[77] Yili Gong, Marlon E. Pierce, and Geoffrey C. Fox. Dynamic Resource-Critical Workflow Scheduling in Heterogeneous Environments. In Eitan Frachtenberg and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, pages 1–15. Springer Berlin Heidelberg, 2009.

[78] Christina Hoffa, Gaurang Mehta, Tim Freeman, Ewa Deelman, Kate Keahey, Bruce Berriman, and John Good. On the Use of Cloud Computing for Scientific Workflows. In *4th IEEE International Conference on eScience (ESCIENCE)*, pages 640–645, 2008.

[79] Ningning Hu and P. Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, Aug 2003.

[80] C. Höfer and G. Karagiannis. Cloud computing services: taxonomy and comparison. *Journal of Internet Services and Applications*, 2:81–94, 2011.

[81] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. In *IEEE 2nd International Conference on Cloud Computing Technology and Science*, pages 159–168, Nov 2010.

[82] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Transactions on Networking*, 11(4):537–549, Aug 2003.

[83] Raj Jain, Dah-Ming Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, Washington University in Saint Louis, 1984.

[84] Q. Jiang, Y. C. Lee, and A. Y. Zomaya. Executing Large Scale Scientific Workflow Ensembles in Public Clouds. In *44th International Conference on Parallel Processing (ICPP)*, pages 520–529, Sept 2015.

[85] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling. Scientific workflow applications on Amazon EC2. In *IEEE International Conference on E-Science Workshops*, pages 59–66, Dec 2009.

[86] G. Juve, M. Rynge, E. Deelman, J. S. Vöckler, and G. B. Berriman. Comparing FutureGrid, Amazon EC2, and Open Science Grid for Scientific Workflows. *Computing in Science Engineering*, 15(4):20–29, July 2013.

[87] Gideon Juve, Ann Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Characterizing and Profiling Scientific Workflows. *Future Gener. Comput. Syst.*, 29(3):682–692, March 2013.

[88] Gideon Juve, Ewa Deelman, G. Bruce Berriman, Benjamin P. Berman, and Philip Maechling. An Evaluation of the Cost and Performance of Scientific Workflows on Amazon EC2. *J. Grid Comput.*, 10(1):5–21, mar 2012.

[89] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P. Berman, and Philip J. Maechling. Data Sharing Options for Scientific Workflows on Amazon EC2. In *22nd IEEE/ACM Conference on Supercomputing (SC10)*, 2010.

[90] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE Int'l Conf. on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[91] Katrina LaCurts, Shuo Deng, Ameesh Goyal, and Hari Balakrishnan. Choreo: Network-aware Task Placement for Cloud Applications. In *Conference on Internet Measurement Conference*, pages 191–204, 2013.

[92] Ang Li, Xiaowei Yang, Srikanth Kandula, and Ming Zhang. CloudCmp: Comparing Public Cloud Providers. In *10th ACM SIGCOMM Conference on Internet Measurement*, pages 1–14, 2010.

[93] K. Li, X. Tang, B. Veeravalli, and K. Li. Scheduling Precedence Constrained Stochastic Tasks on Heterogeneous Cluster Systems. *IEEE Transactions on Computers*, 64(1):191–204, Jan 2015.

[94] Cui Lin and Shiyong Lu. Scheduling Scientific Workflows Elastically for Cloud Computing. In *2011 IEEE International Conference on Cloud Computing (CLOUD)*, pages 746–747, 2011.

[95] Jonathan Livny, Hidayat Teonadi, Miron Livny, and Matthew K. Waldor. High-Throughput, Kingdom-Wide Prediction and Annotation of Bacterial Non-Coding RNAs. *PLOS ONE*, 3(9):1–12, 09 2008.

[96] Vickie Lynch, Jose Borreguero Calvo, Ewa Deelman, Rafael Ferreira da Silva, Monojoy Goswami, Yawei Hui, Eric Lingerfelt, and Jeffrey Vetter. Distributed Workflows for Modeling Experimental Data. In *IEEE High Performance Extreme Computing Conference (HPEC'17)*, 2017.

[97] Maciej Malawski, Gideon Juve, Ewa Deelman, and Jarek Nabrzyski. Algorithms for Cost- and Deadline-Constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds. *Future Generator Computer Systems*, 48:1–18, 2015.

[98] Giacomo V. Mc Evoy and Bruno Schulze. Using Clouds to Address Grid Limitations. In *6th International Workshop on Middleware for Grid Computing (MGC)*, pages 11:1–11:6. ACM, 2008.

[99] Peter M. Mell and Timothy Grance. SP 800-145. The NIST Definition of Cloud Computing. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2011.

[100] Hassan Nawaz, Gideon Juve, Rafael Ferreira da Silva, and Ewa Deelman. Performance Analysis of an I/O-Intensive Workflow executing on Google Cloud and Amazon Web Services. In *18th Workshop on Advances in Parallel and Distributed Computational Models (APDCM)*, pages 535–544, 2016.

[101] A. M. Oprescu and T. Kielmann. Bag-of-Tasks Scheduling under Budget Constraints. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 351–359, Nov 2010.

[102] Simon Ostermann, Alexandria Iosup, Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema. *A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing*, pages 115–131. Springer Berlin Heidelberg, 2010.

[103] Quan-Ke Pan, M. Fatih Tasgetiren, and Yun-Chia Liang. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, 35(9):2807 – 2839, 2008.

[104] S. Pandey, Linlin Wu, S.M. Guru, and R. Buyya. A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. In *IEEE International Conference on Advanced Information Networking and Applications (IANA)*, pages 400 –407, april 2010.

[105] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[106] Giuseppe Papuzzo and Giandomenico Spezzano. Autonomic management of workflows on hybrid Grid-Cloud infrastructure. In *7th International Conference on Network and Service Management (CNSM)*, pages 1–4, oct. 2011.

[107] W. F. Pereira, L. F. Bittencourt, and N. L. S. da Fonseca. Scheduler for data-intensive workflows in public clouds. In *2nd IEEE Latin American Conference on Cloud Computing and Communications (LatinCloud)*, pages 41–46, Dec 2013.

[108] Valerio Persico, Pietro Marchetta, Alessio Botta, and Antonio Pescapè. Measuring network throughput in the cloud: The case of Amazon EC2. *Computer Networks*, 93(P3):408–422, Dec 2015.

[109] Cynthia Phillips, Clifford Stein, and Joel Wein. *Task scheduling in networks*, pages 290–301. Springer Berlin Heidelberg, 1994.

[110] I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou. Energy-constrained provisioning for scientific workflow ensembles. In *3rd International Conference on Cloud and Green Computing (CGC)*, pages 34–41, Sept 2013.

[111] Ilia Pietri, Gideon Juve, Ewa Deelman, and Rizos Sakellariou. A Performance Model to Estimate Execution Time of Scientific Workflows on the Cloud. In *9th Workshop on Workflows in Support of Large-Scale Science (WORKS)*, 2014. Funding Acknowledgements: DOE contract for dv/dt ER26110.

[112] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, 2008.

[113] Riccardo Poli. Analysis of the Publications on the Applications of Particle Swarm Optimisation. *J. Artif. Evol. App.*, 2008:4:1–4:10, jan 2008.

[114] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and K. Ramamohanarao. Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds. In *IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 858–865, May 2014.

[115] R. P. Prado, S. García Galán, A. J. Yuste, J. E. Expósito, A. J. Santiago, and S. Bruque. Evolutionary Fuzzy Scheduler for Grid Computing. In *10th International Work-Conference on Artificial Neural Networks: Part I: Bio-Inspired Systems: Computational and Ambient Intelligence*, pages 286–293, 2009.

[116] Costin Raiciu, Mihail Ionescu, and Dragoş Niculescu. Opening Up Black Box Networks with CloudTalk. In *4th USENIX Conference on Hot Topics in Cloud Ccomputing*, HotCloud'12, pages 6–6, 2012.

[117] Maria A. Rodriguez and Rajkumar Buyya. Budget-Driven Scheduling of Scientific Workflows in IaaS Clouds with Fine-Grained Billing Periods. *ACM Trans. Auton. Adapt. Syst.*, 12(2):5:1–5:22, May 2017.

[118] Maria Alejandra Rodriguez and Rajkumar Buyya. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurrency and Computation: Practice and Experience*, 29(8):e4041–n/a, 2017.

[119] Mats Rynge, Gideon Juve, Jamie Kinney, John Good, G. Bruce Berriman, Ann Merrihew, and Ewa Deelman. Producing an Infrared Multiwavelength Galactic Plane Atlas using Montage, Pegasus and Amazon Web Services. In *23rd Annual Astronomical Data Analysis Software and Systems Conference*, 2013.

[120] Rizos Sakellariou and Henan Zhao. A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems. *Sci. Program.*, 12(4):253–262, December 2004.

[121] Sumit Sanghrajka, Nilesh Mahajan, and Radu Sion. Cloud performance benchmark series – network performance: Amazon EC2. Technical report, Stony Brook University, 2011.

[122] Sumit Sanghrajka and Radu Sion. Cloud Performance Benchmark Series – Network Performance: Rackspace.com. Technical report, Stony Brook University, 2011.

[123] Jörg Schad, Jens Dittrich, and Jorge-Arnulfo Quiané-Ruiz. Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance. *Proc. VLDB Endow.*, 3(1-2):460–471, Sep 2010.

[124] A. Shriram and J. Kaur. Empirical Evaluation of Techniques for Measuring Available Bandwidth. In *26th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2162–2170, May 2007.

[125] Alok Shriram, Margaret Murray, Young Hyun, Nevil Brownlee, Andre Broido, Marina Fomenkov, and kc claffy. Comparison of Public End-to-end Bandwidth Estimation Tools on High-speed Links. In *6th International Conference on Passive and Active Network Measurement*, pages 306–320, 2005.

[126] Joel Sommers, Paul Barford, and Walter Willinger. Laboratory-based Calibration of Available Bandwidth Estimation Tools. *Microprocess. Microsyst.*, 31(4):222–235, Jun 2007.

[127] Georgios L. Stavrinides, Francisco Rodrigo Duro, Helen D. Karatza, Javier Garcia Blas, and Jesus Carretero. Different aspects of workflow scheduling in large-scale distributed systems. *Simulation Modelling Practice and Theory*, 70:120 – 134, 2017.

[128] Xiaoyong Tang, Kenli Li, Guiping Liao, Kui Fang, and Fan Wu. A stochastic scheduling algorithm for precedence constrained tasks on grid. *Future Gener. Comput. Syst.*, 27(8):1083–1091, October 2011.

[129] Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields. *Workflows for e-Science: Scientific Workflows for Grids*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[130] H. Topcuoglu, S. Hariri, and Min-You Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, Mar 2002.

[131] S. S. Vadhiyar and J. J. Dongarra. A performance oriented migration framework for the grid. In *IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 130–137, May 2003.

[132] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers. OpenNetMon: Network monitoring in openflow software-defined networks. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8, May 2014.

[133] Luis M. Vaquero and Luis Rodero-Merino. Finding Your Way in the Fog: Towards a Comprehensive Definition of Fog Computing. *SIGCOMM Comput. Commun. Rev.*, 44(5):27–32, October 2014.

[134] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A Break in the Clouds: Towards a Cloud Definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, December 2008.

[135] C. Vecchiola, S. Pandey, and R. Buyya. High-performance cloud computing: A view of scientific applications. In *10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN)*, pages 4–16, Dec 2009.

[136] Jens-Sönke Vöckler, Gideon Juve, Ewa Deelman, Mats Rynge, and Bruce Berriman. Experiences Using Cloud Computing for a Scientific Workflow Application. In *2nd International Workshop on Scientific Cloud Computing*, pages 15–24, 2011.

[137] Guohui Wang and T. S. Eugene Ng. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *IEEE Conf. on Information Communications (INFOCOM)*, pages 1163–1171, 2010.

[138] Marek Wieczorek, Radu Prodan, and Thomas Fahringer. Scheduling of Scientific Workflows in the ASKALON Grid Environment. *SIGMOD Rec.*, 34(3):56–62, September 2005.

[139] Rich Wolski, Neil T. Spring, and Jim Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Future Gener. Comput. Syst.*, 15(5-6):757–768, October 1999.

[140] Fuhui Wu, Qingbo Wu, and Yusong Tan. Workflow scheduling in cloud: a survey. *The Journal of Supercomputing*, 71(9):3373–3418, 2015.

[141] Yonglei Yao, Jingfa Liu, and Li Ma. Efficient Cost Optimization for Workflow Scheduling on Grids. In *International Conference on Management and Service Science (MASS)*, pages 1 –4, aug. 2010.

[142] U. Yildiz, A. Guabtni, and A. H. H. Ngu. Business versus Scientific Workflows: A Comparative Study. In *4th Congress on Services (SERVICES)*, pages 340–343, July 2009.

[143] J. Yu and R. Buyya. A budget constrained scheduling of workflow applications on utility Grids using genetic algorithms. In *Workshop on Workflows in Support of Large-Scale Science*, pages 1–10, June 2006.

[144] Jia Yu, R. Buyya, and Chen Khong Tham. Cost-based scheduling of scientific workflow applications on utility grids. In *1st International Conference on e-Science and Grid Computing (e-Science)*, pages 8 pp.–147, July 2005.

[145] Z. Yu and W. Shi. A Planner-Guided Scheduling Strategy for Multiple Workflow Applications. In *International Conference on Parallel Processing - Workshops*, pages 1–8, Sept 2008.

[146] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches. *ACM Comput. Surv.*, 47(4):1–33, July 2015.

[147] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2010.

[148] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu. Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing. In *5th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, Sept 2009.

[149] Henan Zhao and R. Sakellariou. Scheduling multiple DAGs onto heterogeneous systems. In *20th IEEE International Parallel Distributed Processing Symposium (IPDPS)*, page 14 pp, April 2006.

[150] A. Zinnen and T. Engel. Deadline constrained scheduling in hybrid clouds with Gaussian processes. In *2011 International Conference onHigh Performance Computing and Simulation (HPCS)*, pages 294 –300, july 2011.

Appendix A

Simulator as a Calculator of Scheduling Events

This thesis employs a simulator as a *calculator of schedule events* to compare schedules produced by several schedulers in order to validate the results. The simulator/calculator is *not* a traditional event-oriented simulator, but it type of a calculator of scheduling events that *simulates* the execution of the workflow according to the schedule given as an input. As this thesis analises schedules produced by different types of scheduling algorithms, the simulator was developed in a way to have a fair analysis comparison among all the produced schedules. The goal of the simulator is to calculate metrics for the scheduling procedure according to the schedule given, i.e., the execution time of each task, the transmission time of each file, the overall monetary execution cost, and the makespan of the workflow(s). By having all this computation, the simulator is capable to calculate, for example, the makespan of the workflow (by checking the finish execution time of the last task of the workflow) as well as the monetary cost of the workflow execution (by carrying out the sum of multiplications of the usage time of each resource by their cost per time). The computation results generated by the simulator strictly follow the schedule given. For example, it registers the start and finish execution time of a task X on a machine/resource Y (as stated in the schedule) as soon as X is "ready" to be executed, i.e., by being ready means that X has received all the necessary files from its antecessors and all the transmissions have already been finalised. This simulator/calculator was developed in 2012 as part of the studies taken in the course *MO648/MC962 Projeto de Redes Multimídia* given by Prof. Dr. Nelson Luis Saldanha da Fonseca at the Institute of Computing of the University of Campinas.

The simulator requires three files as inputs: a schedule file description, a workflow file description, as well as a resource file description. A schedule file description contains the schedule produced by the scheduler, which basically states in which resource each task of the workflow will be executed. The workflow file description contains details of the workflow, such as the number of tasks, the number of files to be transmitted among tasks, the size of each task, as well as the size of each file. In this file, tasks are labeled with their computation cost, millions of instructions (MI) for instance, while files are labelled with their communication cost, bytes to transmit, for instance. Lastly, the resource file description specifies how the resources are grouped and describe their

processing capacities. For example, this file shows the resource processing capacity of each resource (in MI per second (MI/s)), as well as indicates the bandwidths in links connecting the resources (in megabits per second (Mbps)) furnished by the networking tools. By having these there files, Algorithm 13 returns the desired metric for the schedule computation, such as, makespan and cost.

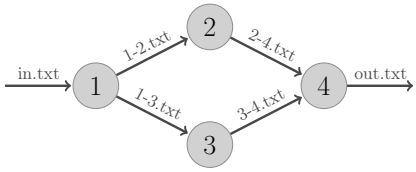


Figure A.1: Workflow representation as a directed acyclic graph (DAG)

Listing A.1 shows an example of the workflow description of the DAG of Figure A.1 that is used as an input to the simulator. The file includes the description of each task and each edge of the workflow according to the DAG that represents it. This representation is based on the DAX file format from Pegasus [22].

```
#Description of the workflow based on DAX file notation of Pegasus
#Description of  files <FILE filename file_size>
FILE in.txt 0
FILE 1-2.txt 100
FILE 1-3.txt 100
FILE 2-4.txt 300
FILE 3-4.txt 300
FILE out.txt 0

#Description of workflow tasks:  <TASK task_id task_name task_size>
TASK ID001 task01 10.0
TASK ID002 task02 60.0
TASK ID003 task03 60.0
TASK ID004 task04 10.0

#Description of workflow edges (task depedencies)  <EDGE task_id task_id>
EDGE ID001 ID002
EDGE ID001 ID003
EDGE ID002 ID004
EDGE ID003 ID004

#Task's outputs/inputs: <INPUTS task_id filename> and <OUTPUTS task_id filename>
#TASK 1
INPUTS ID001 in.txt
OUTPUTS ID001 1-2.txt
OUTPUTS ID001 1-3.txt

#TASK 2
INPUTS ID002 1-2.txt
OUTPUTS ID002 2-4.txt

#TASK 3
INPUTS ID003 1-3.txt
OUTPUTS ID003 3-4.txt

#TASK 4
INPUTS ID004 2-4.txt
INPUTS ID004 3-4.txt
OUTPUTS ID004 out.txt
```

Listing A.1: Workflow file description of the DAG of Figure A.1

An example of the resource file description is shown in Listing A.2. This file is the same used at scheduling time by the scheduler. This file contains information regarding the resources (virtual machines (VMs)) available to execute the workflow. In this file, it is detailed for each VM the following parameters: processing capacities (MIPS), number of processing cores, price per unit time, and the minimum unit time for the VM be charged. Each file represents a cloud provider, thus in case of having a hybrid cloud, several files are used as input to the simulator. The available bandwidth information is also included in the resource file description. Internal bandwidth means the bandwidth value in intra-cloud links, while external bandwidth means the bandwidth value in inter-cloud links.

```
# Description of the VMS for **one** cloud provider  in YAML
# in case of a hybrid cloud, each cloud has your own description file like this one

name: cloud_provider_1

vm-list:

  vm0:
    mips: 2
    cores: 1
    billing:
      unitTime: 1.0
      unitPrice: 0.05

  vm1:
    mips: 4
    cores: 2
    billing:
      unitTime: 1.0
      unitPrice: 0.10

bandwidth:
  internal:
    value: 100.0
    unit: Mbps

  external:
    value: 10.0
    unit: Mbps
```

Listing A.2: Resource file description

Lastly, the schedule file description is depicted in Listing A.3. This file details in which resource of which cloud provider each task of the workflow will be executed. This is file guides the simulator to calculate the necessary metrics for the scheduling procedure, such as the makespan of the workflow. Therefore, the simulator strictly follows this file to simulate the execution of the workflow according to the scheduler's decisions, i.e., to perform the calculation of the scheduling events. Note that, to produce the schedule file description, the scheduler uses as input that the workflow file description and the resource file description detailed above.

```
# Schedule given by the scheduler in YAML
# Informs in which resource (VM) the task where scheduled to run

schedule-list:

  ID001:
    cloud_provider_1
    vm0

  ID002:
    cloud_provider_1
    vm1

  ID003:
    cloud_provider_1
    vm1

  ID004:
    cloud_provider_1
    vm0
```

Listing A.3: Schedule file description

Gathering all input data described above, the simulator performs the calculation of the scheduling events. The general steps of the simulator are presented in Algorithm 13. Guided by the schedule file description, the simulator send for execution on the resource stated on the schedule file only tasks that are classified "ready". A task is *ready* if and only if has received all the necessary files (inputs) from all the task's intercessors and all the transmission has been finalised (Lines 6-15). When the task is ready, the execution is recorded on the log file of the VM that execute it and then all data transmission for all the successors are also recorded on the log file of the VM that is sending and the VM that is receiving (Lines 25-33). The simulator finalises its execution when all the tasks have been executed and all the pending transmission have been finalised (Lines 19-21).

```
# Log
name: cloud_provider_1
vm: vm0

Event              Description        Time
Execute_Task       Task ID001         0 to 5
Send_File          1-2.txt to vm1     5 to 6
Send_File          1-3.txt to vm1     5 to 6
Receive_File       2-4.txt from vm1   21 to 24
Receive_File       3-4.txt from vm1   21 to 24
Execute_Task       Task ID004         24 to 29
=============================================================
Usage time: 29 - 0 = 0
Cost: 29/1.0 * 0.05 = $1.45
```

Listing A.4: Example of the log of $vm0$ produced by the simulator due to the schedule in Listing A.3.

Listings A.4 and A.5 illustrate the log files produced for $vm0$ and $vm1$ produced by the simulator due to the schedule in Listing A.3. The simulator creates for each VM a log file that stores all the schedule events carried by the VM. By reading these log files, the simulator can easily calculate the makespan of the workflow as well as its monetary execution cost. For the schedule example illustrated in Listing A.3, the makespan of the

workflow is 29 and it computation cost is $1.45 + 1.90 = \$3.35$. The source code of the simulator is available at GitHub. For more simulator/calculator details, please, access: `https://github.com/thiagogenez/schedule-simulator`.

```
# Log
name: cloud_provider_1
vm: vm1

Event              Description        Time
Receive_File       1-2.txt from vm0   5 to 6
Receive_File       1-3.txt from vm0   5 to 6
Execute_Task       Task ID002         6 to 21
Execute_Task       Task ID003         6 to 21
Send_File          2-4.txt to vm0     21 to 24
Send_File          3-4.txt to vm0     21 to 24
=============================================================
Usage time: 24 - 5 = 19
Cost: 19/1.0 * 0.1 = \$1.90
```

Listing A.5: Example of the log of $vm1$ produced by the simulator due to the schedule in Listing A.3

Algorithm 13 Simulator/Calculator of Schedule Events Procedure Overview

Require: A schedule produced by a scheduler, a workflow file description, a cloud file description
Ensure: Requested schedules metrics (monetary cost, makespan, speedup, etc)

$\mathcal{Q} \leftarrow$ an empty priority queue for tasks that are "ready" to be executed ▷ *tasks ordered according to the finish execution time given by the schedule*
$\mathcal{T} \leftarrow$ a list that stores the tasks of the workflow

while true **do**

    for all task $t \in \mathcal{T}$ **do**
        $\mathcal{L} \leftarrow$ an empty list
        if has received all the necessary files and all the transmission has been finalised **then**
            $\mathcal{Q}.add(t)$
            $v \leftarrow$ the virtual machine where the task t will be executed according to the schedule
            $v.log.executionList.add(t)$
        else
            $\mathcal{L}.add(t)$
        end if Ss
    end for

    $\mathcal{T} = \mathcal{L}$ ▷ Tasks that have not been executed

    if $\mathcal{Q}$ and $\mathcal{T}$ are empty **then**
        Break ▷ All tasks have been executed and all transmissions have been completed
    end if

    $t \leftarrow \mathcal{Q}.pop$ ▷ Pop from the queue the task with early finish time

    for all successor of t **do** ▷ transmission that t has to carry out
        $succ \leftarrow$ the successor task of t according to the description of the workflow
        $from \leftarrow$ the virtual machine where t will be executed according to the schedule
        $to \leftarrow$ the virtual machine where $succ$ will be executed according to the schedule
        $file \leftarrow$ the file to transmit from t to $succ$

        $from.log.sendFileList(file)$
        $to.log.receiveFileList(file)$
    end for
end while

Check all the VM logs and calculate the requested metric, such as the makespan, monetary cost, speedup, etc
return Requested metrics