MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

DYNAMIC MULTI-OBJECTIVE WORKFLOW SCHEDULING IN CLOUD COMPUTING

GOSHGAR ISMAYILOV

MASTER THESIS
Department of Computer Engineering

Thesis Supervisor
Prof. Haluk Rahmi Topcuoglu

ISTANBUL, 2019

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

DYNAMIC MULTI-OBJECTIVE WORKFLOW SCHEDULING IN CLOUD COMPUTING

GOSHGAR ISMAYILOV

(524116902)

MASTER THESIS
Department of Computer Engineering

Thesis Supervisor
Prof. Haluk Rahmi Topcuoglu

ISTANBUL, 2019

MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN PURE AND APPLIED SCIENCES

Goshgar ISMAYILOV, a Master of Science student of Marmara University Institute for Graduate Studies in Pure and Applied Sciences, defended his thesis entitled "**Dynamic Multi-Objective Workflow Scheduling in Cloud Computing**", on August 8, 2019 and has been found to be satisfactory by the jury members.

Jury Members

Prof. Dr. Haluk Rahmi TOPCUOĞLU (Advisor)
Marmara University .....

Prof. Dr. Can ÖZTURAN (Jury Member)
Boğazici University .....

Assist. Prof. Ömer KORÇAK (Jury Member)
Marmara University .....

APPROVAL

Marmara University Institute for Graduate Studies in Pure and Applied Sciences Executive Committee approves that Goshgar ISMAYILOV be granted the degree of Master of Science in Department of Computer Engineering, Computer Engineering Program on 21.08.2019 (Resolution no: 2019/17-02).

Director of the Institute
Prof. Dr. Bülent EKİCİ

ACKNOWLEDGEMENTS

First, I owe my deepest gratitude to my advisor Prof. Haluk Rahmi Topcuoglu for his continuous support and encouragement for my master thesis, for his endless motivation, and literally commitment to his profession. It has always been my great privilege to be guided by him at every moment of the research.

I would like to thank Prof. Can Özturan and Asst. Prof. Ömer Korçak for participating in my thesis committee and giving me helpful feedbacks.

I wish to acknowledge the support of my dear family who have planted the passion for science in my soul since my childhood; unconditional love of my dear fiancée, Kubra Altun, who has always been with me.

July, 2019 **Goshgar ISMAYILOV**

TABLE OF CONTENTS

ÖZET

BULUT HESAPLAMA İÇİN DİNAMİK ÇOK-AMAÇLI İŞ AKIŞI ÇİZELGELENMESİ

Bulut hesaplama, isteğe bağlı olarak farklı düzeylerde hizmetler sunan, heterojen ve dağıtık bir hesaplama sistemidir. Bulut sistemlerdeki mevcut kaynakların iş akışlarının yürütülmesi için, her bir iş akışının servis kalitesinde belirtilen hedefleri gözeterek etkin kullanımını amaçlayan iş akışı çizelgeleme problemi, üzerinde çokça çalışılmış araştırma problemlerinden biridir. Zaman içerisindeki kaynak arızalanmalarını ve iş akışı amaçlarının zaman içerisindeki değişimini göz önüne alan dinamik iş akışı çizelgeleme problemi, bu tez kapsamında, bir dinamik çok amaçlı eniyileme problemi olarak modellenmiştir. Yazılımsal veya donanımsal hatalar, ilk türdeki dinamik ögeye yönelik başlıca nedenler arasında sayılabilir. Değişen hizmet kalitesi gibi bulut sistemlerinde rastlanan farklı gerçek hayat senaryolarının bir iş akışının yürütülmesi sırasında eniyileme amaçlarının sayısını değiştirebilmesi, ikinci türdeki devingenliğe yönelik temel motivasyon kaynağı olarak düşünülebilir. Bu tezde, dinamik iş akışı çizelgeleme problemi için iki farklı çözüm geliştirilmiştir. İlk çözümde, literatürden seçilen, tahmine dayalı olmayan beş öncü çok-amaçlı evrimsel algoritma, bu problem için uyarlanmıştır. İkincil çözümde ise, çok amaçlı eniyileme problemleri için yoğun olarak kullanılan NSGA-II algoritmasını yapay sinir ağları ile birleştirerek, tahmine dayalı yeni bir dinamik çok amaçlı eniyileme algoritması (NN-DNSGA-II algoritması) önerilmiştir. Bu problem için çözümler, altı farklı amaç fonksiyonu arasındaki ödünleşim göz önünde bulundurularak oluşturulmuştur. Bu amaç fonksiyonları, iş akışının yürütülmesi için gerekli sürenin, maliyetin, enerjinin ve kaynaklar arası iş yükü dengesizliklerinin minimizasyonu, ve güvenilirliğin ve kaynaklardan faydalanma oranının maksimizasyonudur. Pegasus iş akışı yönetim sistemindeki uygulamalar ve Amazon EC2 sistemindeki kaynaklar kullanılarak deneysel çalışma gerçeklenmiştir. NN-DNSGA-II algoritması, domine edilmeyen çözümlerin sayısı, Schott aralığı ve Hiper-hacim metriklerine göre, diğer alternatiflerden belirgin olarak daha iyi sonuçlar verdiği gözlemlenmiştir.

July, 2019 **Goshgar ISMAYILOV**

ABSTRACT

DYNAMIC MULTI-OBJECTIVE WORKFLOW SCHEDULING IN CLOUD COMPUTING

Cloud computing is a heterogeneous and distributed computing platform that delivers various levels of on-demand services to the customers. Workflow scheduling is one of the research challenges, which is largely addressed by cloud computing in the literature, where efficient utilization of cloud resources for workflow executions is targeted by considering the objectives specified in QoS requirements. In this thesis, the dynamic workflow scheduling problem is modelled as a dynamic multi-objective optimization problem (DMOP) where the source of dynamism is based on both resource failures and the number of objectives which may change over time. Software faults and/or hardware faults can be listed among the motivations for the first type of the dynamism. On the other hand, several real-life scenarios confronted in cloud computing such as changing in QoS requirements may change number of objectives at runtime during the execution of a workflow, which is the main motivation for the second type of dynamism. In this thesis, two different solutions are proposed for the dynamic workflow scheduling problem. In the first solution, five non-prediction based multi-objective evolutionary algorithms from the literature are adapted for the problem. In the second solution, a novel prediction-based dynamic multi-objective evolutionary algorithm (NN-DNSGA-II) is presented, where it incorporates neural networks with the NSGA-II algorithm. Trade-offs between six different objectives are considered while scheduling, which are minimization of makespan, cost, energy and degree of imbalance; and maximization of reliability and utilization. The empirical study is based on real-world applications from the Pegasus workflow management system and resource specifications from Amazon EC2. It reveals that our NN-DNSGA-II algorithm significantly outperforms the other alternatives in most cases with respect to three different metrics used for DMOPs with unknown true Pareto-optimal front, consisting of the number of non-dominated solutions, the Schott's spacing and the Hypervolume indicator.

July, 2019 **Goshgar ISMAYILOV**

ABBREVIATIONS

CMOS : Complementary Metal-Oxide Semiconductor
DAG : Directed Acyclic Graph
DMOEA : Dynamic Multi-Objective Evolutionary Algorithm
DMOP : Dynamic Multi-Objective Optimization Problem
DMOPSO : Dynamic Multi-Objective Particle Swarm Optimization
EC2 : Elastic Cloud Computing
EDO : Evolutionary Dynamic Optimization
GA : Genetic Algorithm
HM : Hyper-Mutation
HV : Hypervolume
IaaS : Infrastructure as a Service
MTBF : Mean Time Between Failures
MTTF : Mean Time to Failure
MTTR : Mean Time to Recover
NN : Neural Network
NS : Number of Non-dominated Solutions
NSGA : Non-Dominated Sort Genetic Algorithm
P2P : Peer-to-peer
PaaS : Platform as a Service
POF : Pareto-optimal Front
POS : Pareto-optimal Set
PSO : Particle Swarm Optimization
RM : Random Immigrants
SaaS : Software as a Service
SLA : Service Level Agreement
SS : Schott's Spacing
SVM : Support Vector Machine
QoS : Quality of Service
VM : Virtual Machine

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Cloud computing is a large-scale heterogeneous and distributed computing paradigm for the scientific and commercial communities, where high quality and low cost services are provided with minimal hardware investments. The well-known service layers cloud computing delivers over the internet are PaaS, SaaS and IaaS. In this thesis, Infrastructure-as-a-Service (IaaS) is mostly referred, where the customers can access hardware resources, on which the applications can be deployed. The requirements of customers from clouds refer to Quality-of-Services (QoS) parameters, which is defined under the assurance of the Service Level Agreement (SLA). In return for the cloud services, the customers are charged with respect to the duration of their usages and QoS parameters.

The advantageous attributes of cloud computing has been leading to the proliferation of many cloud infrastructures around the world in recent years, including Microsoft Azure, Google Cloud and Amazon EC2. The virtualization technology, the elasticity and pay-as-you-go pricing scheme are among those key attributes. The virtualization technology in IaaS enables the available on-demand resources to be efficiently provisioned in the form of virtual machines (VM). It enables multiple applications with different requirements to run in the same physical resource with time and hardware sharing principles. The elasticity offers the customers an extra flexibility, where the number of resources is dynamically scalable with respect to the requirements of their applications and budgets. Lastly, pay-as-you-go pricing strategy charges on the basis of number of time frames, mostly hours, which helps to reduce both the execution time and cost.

Workflows are among the techniques that are frequently used to construct large scale compute-intensive and data-intensive applications from different research domains. An application workflow is modelled with a directed acyclic graph where the nodes of the graph are precedence constrained tasks that are interconnected via resources in the system. Those workflows are mostly large and complex, which requires massive amount of resources for their executions. Therefore, their deployments on cloud computing is generally considered as more favourable. However, the adoption of cloud computing for workflows leads to certain and unique problems. The workflow scheduling problem

in cloud computing is among them, where it mainly aims to map the tasks of a given application onto available resources [1, 2, 3]. It is known as an NP-complete problem [1]. The efficient and automatic orchestration and the management of task executions in the workflows are the key concerns in the problem to optimize the given objectives in QoS under certain constraints.

The periodical workflow scheduling mostly involves with the applications that are periodically carried out and submitted to the system, where those applications have been increasingly encountered in different domains of science and industry. They emerge in physics for gravitational waves yearly [4], in business for fiscal analysis of companies monthly and in meteorology for weather forecasting daily and storm surge and wind wave prediction hourly in life-threatening extreme conditions [5]. The workloads of the tasks in the applications may vary in every single period according to the amount of data they collect. These unpredictable workloads fluctuations lead to period-based scheduling where the tasks should be rescheduled with respect to their latest workloads in every period.

The concept of dynamism in workflow scheduling in this study is twofold. The first type of dynamism is transient resource failures over time, where the resources may dynamically join to and leave from the cloud. They may arise in consequence of several events such as software faults (bugs, overflows, etc.) or hardware faults (irregular electric power, hard disk failures, etc.). The other source for dynamism in workflow scheduling problem is changing number of objectives during execution of workflows. Cloud computing confronts with real-life scenarios, where the number of objectives may change over time [6]. For instance, makespan of a workflow may not be taken into the consideration until a workflow with tighter deadline is submitted for execution. It is also emphasized in the literature that cloud computing is one of the subjects that is open for more variety of changing objectives to be considered. The objectives can be ignored or considered by several external factors including the level of power consumption, the level of imbalance of workloads among resources or changing workflow QoS requirements. The number of objectives and the interactions between the objectives considered in this dynamism have impacts on the selection of optimal workflow scheduling solutions in different periods.

To the best of our knowledge, our study is among the first attempts to model the workflow scheduling problem in cloud computing as a dynamic multi-objective optimization problem (DMOP) by considering the resource failures and changes in the number of objectives as main sources of dynamism. The dynamic multi-objective evolutionary algorithms (DMOEAs) are utilized in order to solve the existing problem at the end by efficiently coping with these dynamic characteristics of the problem. A new prediction-based dynamic multi-objective evolutionary algorithm is proposed (the NN-DNSGA-II algorithm), that incorporates neural network with NSGA-II algorithm. The NN-DNSGA-II algorithm combines the strength of the neural network with dynamic NSGA-II algorithm to reveal the change or the movement patterns of the optimal solutions in the optimization environments. It exploits the correlations between task-resource pairs and the correlations between two successive optimization environments so that the future positions of optimal solutions can be estimated based on their past positions.

In the scope of this thesis, five multi-objective evolutionary algorithms are adapted from the literature that do not need any prediction mechanism. They include the DNSGA-II-A, the DNSGA-II-B, the DNSGA-II-HM, the DNSGA-II-RI algorithms and the dynamic variation of the particle swarm optimization algorithm called the DMOPSO algorithm. They are especially considered for the types of changes in the scheduling scenarios that are not prone to be predictable. Each of them has different type of change response mechanisms to cope with the environmental dynamism. For instance, the DNSGA-II-A algorithm, the DMOPSO and the DNSGA-II-RI algorithms basically insert random solutions to re-diversify the population. While the former two algorithms need to detect the changes at first to run their response mechanisms, the latter one tries to preserve certain level of random solutions in the population without detecting the changes. The DNSGA-II-B algorithm inserts mutated versions of the existing solutions in the population. Finally, the DNSGA-II-HM algorithm uses adaptation of mutation rates in the environment.

An extensive empirical study is carried out among the algorithms, where the optimization objectives are the minimization of *makespan*, *cost*, *energy* and *degree of imbalance*; and the maximization of *reliability* and *utilization*. The specifications details of the resources are based on real system which is Amazon EC2. The workflows that

are executed on this system as test beds are real workflow applications that are taken from Pegasus Workflow Management System. The comparisons of the algorithms are carried out by three different metrics that are mostly used for dynamic multi-objective optimization problems, which are the number of non-dominated solutions, Schott's spacing and Hypervolume. For the first type of dynamism in this thesis, the effect of frequency of resource failures, severity of resource failures, varying number of resources and varying variance of uncertainties in the system are measured on the performance of the algorithms. For the second type of dynamism, The effect of changing number of objectives at each scheduling period, the frequency and the severity of changing number of objectives are measured on the performance of the algorithms. The experimental evaluation validates that our NN-DNSGA-II algorithm outperforms the adapted algorithms up to 24 out of 30 cases and 35 out of 45 cases with respect to frequency and severity of resource failures, respectively; and up to 25 out of 30 cases and 22 out of 30 cases with respect to frequency and severity of changing number of objectives.

1.1. Contributions of the Thesis

The key contributions of our thesis can be outlined as follows:

- The dynamic workflow scheduling problem in cloud computing is defined as a dynamic multi-objective optimization problem, where the dynamism are resource failures and changing number of objectives.
- Five state-of-art multi-objective evolutionary algorithms (the DNSGA-II-A, the DNSGA-II-B, the DNSGA-II-HM, the DNSGA-II-RI and the DMOPSO algorithms) are adapted to the given problem from the literature.
- A new prediction-based dynamic multi-objective evolutionary algorithm that incorporates the NSGA-II algorithm with neural networks (NN-DNSGA-II) is proposed for the given problem.
- The characteristics of a cloud computing system such as delays and uncertainties are also taken into the consideration while scheduling the workflows. Six different optimization objectives are considered, which are the minimization of makespan cost, energy and degree of imbalance, and the maximization of reliability and utilization. Based on real cloud computing system and real workflow applications,

the experimental study is carried out, where the NN-DNSGA-II algorithm outperforms the other algorithms in the most cases of the experiments for the resource failures and changing number of objectives.

1.2. Outline of the Thesis

The rest of the thesis is organized as follows. In Section 2, the workflow scheduling problem in cloud computing is presented, which includes application model, system architecture model, reliability model, energy model, the optimization objectives considered during the workflow execution and finally cloud-based delays and uncertainties. In Section 3, general overview of dynamic multi-objective optimization is presented. The dynamic multi-objective optimization techniques and their categorizations to solve dynamic multi-objective optimization problems are also explained. In Section 4, the details of dynamic workflow scheduling problem and two different proposed solutions to that problem are presented. Finally, the time complexity analysis of all the algorithms is provided. Section 5 presents experimental setup including the real-world workflows from different domains of science, the real-world cloud computing infrastructure based on Amazon EC2 and settings for any other existing parameters. It also defines three different performance metrics to compare the performance of the algorithms in this thesis. Section 6 presents the results and discussion for the failures of resources in the system and changing number of optimization objectives. Finally, Section 7 concludes the paper and presents the future work.

2. WORKFLOW SCHEDULING PROBLEM IN CLOUD COMPUTING

In this section, the application, system, reliability and energy models are described for the dynamic workflow scheduling problem in cloud system. The six different optimization objectives for the problem have been presented in this section as well. They include the minimization of makespan, cost, energy and degree of imbalance; and the maximization of reliability and utilization.

2.1. Application Model

A workflow application, $G = (T, D)$, is modelled through a directed acyclic graph with no cycles and conditional dependencies, where $T = (t_1, t_2, ..., t_n)$ is the set of tasks and $D = \left(d_{ij} \mid t_i, t_j \in T\right)$ is the set of edges among precedence-constrained parent and child tasks. The weight of a task indicates its reference execution time in seconds and the weight of an edge indicates its communicational output data size in bytes. The output data that is generated by parent task is required to be transmitted to child task for the continuity of the workflow execution. For a task t_i, $pred(t_i) = \{t_j | d_{ji} \in D\}$ refers the set of its immediate predecessors and $succ(t_i) = \{t_j | d_{ij} \in D\}$ refers the set of its immediate successors. A task with no predecessor is labelled as $t_{entry} = \{t_i | pred(t_i) = \emptyset\}$ and a task with no successor is labelled as $t_{exit} = \{t_i | succ(t_i) = \emptyset\}$. Note that Table 2.1. lists all the key notations used through thesis.

In this study, it is assumed that tasks are not further decomposable. Therefore, each of them can only be scheduled to a single resources in a cloud system. On the other hand, a resource can execute multiple tasks during its billing period by processing at most one task at a time. The execution of a child task can be only realized after all its parent tasks have been successfully completed and their output data is successfully transferred to the child task. The execution of a task proceeds with non-preemptive manner, which means that once a task starts its execution on a resource, it cannot be interrupted any more by any other external events except from the failure of that resource.

Any two tasks in a workflow that share single unique parent and single unique child relationship in sequence are considered as pipelined. The tasks in pipeline are merged as single task by vertical clustering in our study. The operation first recursively traverses all

Table 2.1: Notations

Symbol	Definition
$RT(t_i)$	The reference time of task t_i
$ST(t_i, r_k)$	The start time of task t_i on resource r_k
$PT(t_i, r_k)$	The processing time of task t_i on resource r_k
$TT(t_i, t_j)$	The transfer time of task t_i and task t_j
$FT(t_i, r_k)$	The finish time of task t_i on resource r_k
CU_k	Computing unit of resource r_k
C_k	Cost of resource r_k
N_k	Number of billing period that r_k is provisioned
B	System bandwidth
DL	Delay
DI	Degree of imbalance
R	Reliability
E	Energy consumption
τ	Billing period

the tasks in a graph until there is no more tasks in a pipeline exist and then scheduling period of the resulting clustered tasks to resources begins. The operation is used to yield improvement over the execution performance of baseline workflow scheduling by eliminating the transmission overhead of both time and money in this study.

The periodical workflows are certain types of applications whose tasks arrive at the system periodically. The reference execution times (the weights of the tasks) and communicational output data size (the weights of the edges) may naturally fluctuate to certain extents based on the amount of information collected at each periodical stage. The stochastic behaviour in both of these components of the applications due to unpredictable variations increases the complexity of the scheduling problem. Such variations necessarily leads to periodical workflow scheduling conception, in which the tasks of the workflows need to be rescheduled in every separate period with respect to latest information about their workloads.

2.2. System Model

Let $R = (r_1, r_2, ..., r_m)$ represent a set of heterogeneous computational resources. Our computing model is consisted of single IaaS cloud provider with single data center.

A workflow scheduler is responsible of generating scheduling solutions and assigning the tasks in the workflows in correct orders to correct resources and set of resources. Our IaaS platform offers a range of resource types with different combinations of resource attributes such as and computing unit (CU) and price. The first attribute, computing unit of a resource, is used as a relative indicator to predict and compare its performance with performances of other resources. The second attribute, price of a resource, is based on billing period for provisioning. Any partial billing period in the system are billed as full billing period. The resources with higher computing unit are intrinsically charged by higher prices. Each resource can contain only one virtual machine (VM) at a time. In this regard, the processing time of task t_i on resource r_k is calculated as follows:

$$PT(t_i, r_k) = \frac{RT(t_i)}{CU_k} \tag{2.1}$$

The resources are part of fully inter-connected mesh topology, where direct links between resources based on peer-to-peer (P2P) paradigm ensure the output data transmission to be freely performed without contention. The transfer time between two precedence-constrained tasks depends on amount of output data transferred and transmission bandwidth, which is assumed as identical. The intra-resource transmission occur in the local of single resource while the inter-resource transmission involves with two different resources. When compared with the inter-resource transmission, the intra-resource transmission is minimal and negligible. The transfer time between two dependent tasks t_i and t_j over a link with bandwidth B is calculated as follows:

$$TT(t_i, t_j) = \begin{cases} \frac{d_{ij}}{B} & r_k \neq r_{k'} \\ 0 & r_k = r_{k'} \end{cases} \tag{2.2}$$

The start and finish times of the tasks in the workflows are recursively defined with respect to the processing times of the tasks and transfer times of output data. These recurrence relations for the task executions are calculated as follows:

$$ST(t_{entry}, r_k) = ready(r_k) \tag{2.3}$$

$$ST(t_i, r_k) = max(ready(r_k), \max(FT(t_p, r_{k'})) \tag{2.4}$$

$$FT(t_i, r_k) = ST(t_i, r_k) + PT(t_i, r_k) + \sum_{t_c \in succ(t_i)} TT(t_c, t_i) \tag{2.5}$$

where $t_p \in pred(t_i)$, $t_c \in succ(t_i)$ and $ready(r_k)$ shows the time resource r_k is ready to execute the next task. A resource is considered as ready for execution when it is provisioned and its boot-up ends if it is not provisioned at the time; or it is free from any task if it is provisioned at the time.

The objectives that are considered in our study includes the minimization of *makespan*, *cost*, *energy* and *degree of imbalance*; and the maximization of *reliability* and *utilization* while satisfying the precedence constraints between tasks. The set of solutions schemes which take the trade-offs between these objectives are taken into the account. The formulations of the four objectives (makespan, cost, utilization, degree of imbalance) are given in the following four subsections, while the formulations of two remaining ones (reliability and energy) are given in separate sections.

2.2.1. Makespan

The first objective of dynamic workflow scheduling problem in cloud computing, *makespan*, can be defined as minimization of maximum finish time of the task among all the tasks as follows:

$$\max_{t_i \in T} (FT(t_i, r_k)) \tag{2.6}$$

2.2.2. Cost

In this study, a complete cost model is built, which considers the cost of task execution, output data transmission and output data storage. The cost of output transmission between two dependent tasks that are executed in the same resource (intra-resource communication) is neglected. The cost of execution of a workflow in cloud computing is defined as follows:

$$\left(\sum_{r_k \in R} N_k \times C_k^{execution}\right) + \left(\sum_{d_{ij} \in D} d_{ij} \times C_k^{transfer}\right) + \left(\sum_{d_{ij} \in D} d_{ij} \times time(d_{ij}) \times C_k^{storage}\right) \tag{2.7}$$

where $C_k^{execution}$,$C_k^{transfer}$ and $C_k^{storage}$ are the execution, transmission and storage prices of the resource r_k, respectively; and $time(d_{ij})$ shows the amount of time the output data is stored, which spans from the first production of output data until the end of workflow. The units of the execution, transmission and storage prices are dollars per billing period ($\$/hr$), dollars per transmission of one gigabyte output data ($\$/GB$) and dollars per storage of one gigabyte output data per one hour ($\$/GB/hr$). The complete model helps to obtain more accurate results when involving with a data intensive workflow.

2.2.3. Utilization

Utilization is another key issue for high efficiency and performance. The selection of optimal number of resources while scheduling is directly associated with the utilization. For instance, under-provisioning (i.e. provisioning less resources than actually needed) may yield high utilization rate, but it may yield greater makespan at the same time. On the other hand, over-provisioning may yield lower utilization, but it may yield smaller makespan. The *utilization* as ratio of total amount of time that the resources are busy to total amount of time that the resources are serviceable is defined as follows:

$$\frac{\sum_{t_i \in T} PT(t_i, r_k)}{\sum_{r_k \in R}(\tau - DL_{prv} - DL_{deprv}) \times N_k} \tag{2.8}$$

where DL_{prv} and DL_{deprv} denote the provisioning and deprovisioning delays, respectively. They are removed from the formulation in order to find the net *utilization*. The multiplicative inverse of utilization is considered in calculations for ease of use.

2.2.4. Degree of Imbalance

The degree of imbalance focuses on the imbalance of task workloads among all available resources. The consideration of utilization only without the degree of imbalance among resources may mislead to incorrect assessments about performance of cloud

system. It is possible to have low degree of imbalance even if the utilization rate of the system is quite high. The degree of imbalance of an individual resource r_k is calculated as follows:

$$DI_k = \sum_{t_i \in T} PT(t_i, r_k) \tag{2.9}$$

where DI_k is the summation of processing times of the tasks that are executed on the resource r_k. Finally, the degree of imbalance of total system is defined as follows:

$$\frac{DI_{max} - DI_{min}}{DI_{avg}} \tag{2.10}$$

where DI_{min} and DI_{max} represent the minimum and the maximum degree of imbalance values of individual resources, respectively; and $DI_{avg} = \frac{1}{m} \sum_{r_k \in R} DI_k$ represents the average degree of imbalance among the resources. Finally, a resource that already receives the request and starts to shut down itself cannot execute any tasks any more. In case execution of a task is still ongoing while billing period ends, an extra billing period of that resource necessarily begins. A resource is provisioned only when there exists at least one ready task that is assigned to that resource by the scheduler. Since price of a resource has to be paid even if its billing period is not fully utilized or there may be other tasks waiting to be executed on that resource in the same billing period, it is shut down only at the end of the billing period.

2.3. Reliability

The failures of the resources are addressed in this study to quantitatively assess the reliability of cloud computing for robust scheduling solutions. Our reliability model is fundamentally based on long-term empirical resource availability data presented in a related work [7]. It reveals the characteristics of resource failures in large scale high computing environments. These characteristics include the mean time between two consecutive failures, the number of resources that fail and the duration of failures. According to the graphical analysis that is presented in the Figure 2.3, Weibull is the fittest distribution to model the inter-arrival time between failures and the size of failures. The two-parameter Weibull distribution with scale (η) and shape (β) parameters is as

follows:

$$p(x) = \frac{\beta x^{\beta-1}}{\eta^{\beta}} e^{-\left(\frac{x}{\eta}\right)^{\beta}} \tag{2.11}$$

where x is a random variable, which represents a point in time and $p(x)$ represents the failure probability at that point in time. The different scale and shape parameter values of Weibull distribution can be used to describe different failure patterns. The effect of scale parameter can be seen in the Figure 2.1, where the increase in scale value stretches the distribution out to the right in time scale and gradually decreases the maximum probability of failure in time. On the other hand, the effect of shape parameter can be seen in the Figure 2.2, where failure probability decreases over time in infant mortality period ($0 < \beta < 1$), remains constant in useful period $\beta = 1$ and increases over time in wear-out period $\beta > 1$.

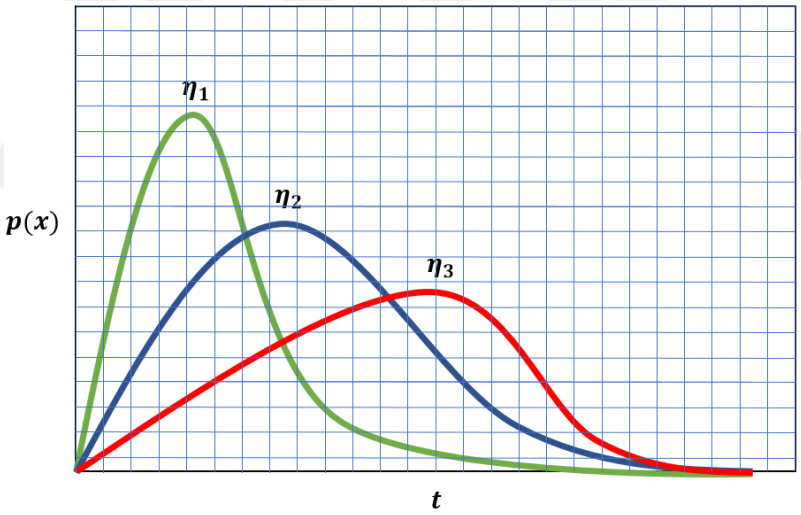


Figure 2.1: The effect of scale parameter in probability density function (PDF) of Weibull distribution with fixed shape parameter, $\eta_1 < \eta_2 < \eta_3$

In our study, the reliability can be defined as the success probability that a resource can carry out its intended functionalities to execute tasks under different customer and system requirements. The success probability and failure probability are complementary to each other and their sum must be equal to one:

$$R(t_i, s_j) + F(t_i, s_j) = 1 \tag{2.12}$$

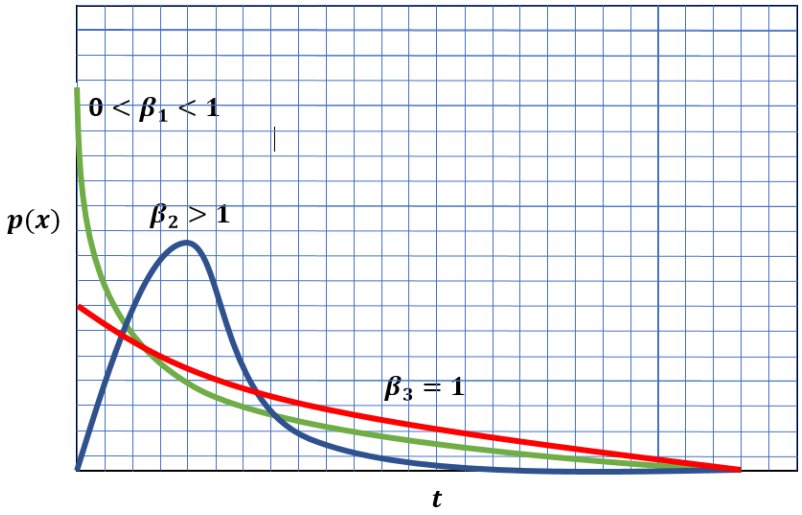
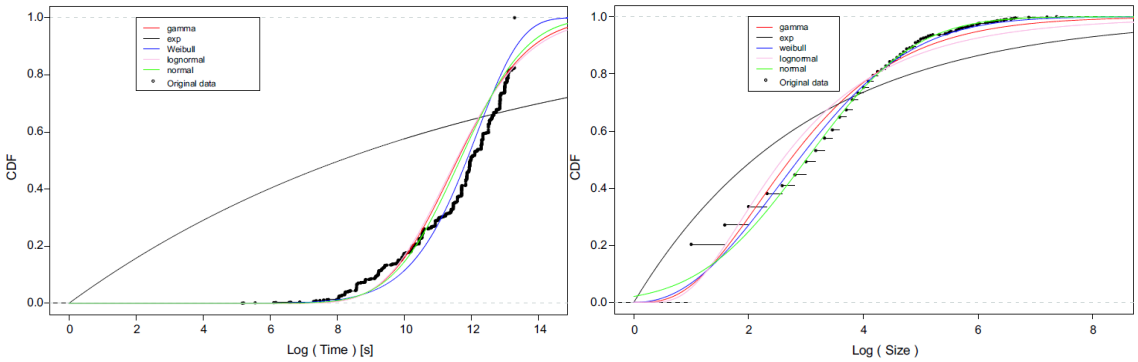


Figure 2.2: The effect of shape parameter in probability density function (PDF) of Weibull distribution with fixed scale parameter, $\beta_1 < \beta_3 < \beta_2$

For a single task, the integration of individual failure probabilities at each time point during its execution interval yields its cumulative likelihood of being not successfully completed. So, reliability of a task which is dependent to its execution time can be defined as follows:

$$R(t_i, r_k) = 1 - \int_{ST(t_i,r_k)}^{FT(t_i,r_k)} p(x)dx = e^{-\left(\frac{PT(t_i,r_k)}{\eta}\right)^{\beta}} \tag{2.13}$$



(a) Inter-arrival time between failures **(b)** Size of failures

Figure 2.3: Characterization of resource failures with cumulative distribution functions (CDF) of various statistical distributions [7]

Then, the multiplicative aggregation of the reliability values for precedence-constrained tasks gives the overall reliability of workflow, which is defined as follows:

$$R(G) = \prod_{t_i \in T} R(t_i, r_k) \tag{2.14}$$

The multiplicative inverse of reliability, $R(G)$, has been used for ease of use. In periodical workflow scheduling, once resources temporarily fail, they cannot participate into that scheduling period. The duration of failures for recovery is comparably smaller than the inter-arrival time between failures [7]. It can be inferred that the resources that fail in previous scheduling period, are capable of taking part in the next period. In the study, the communicational and network components in the system are reliable, bug-free and always available.

2.4. Energy

Dynamic Voltage Frequency Scaling (DVFS) technique enables to control the changes in clock frequency and voltage of a CPU in order to minimize the power consumption. Our DVFS-based energy model is based on power consumption of complementary metal-oxide semiconductor (CMOS) logic circuits. The power consumption of a CMOS model is consisted of three different factors including the dynamic power dissipation, short-circuit and leakage power:

$$P = KV^2 f + I_{leak} V + P_{short} \tag{2.15}$$

where K is the coefficient constant for number of switches per clock cycle and the total capacitance load, V is the supply voltage, f is frequency, I_{leak} is the leakage current and P_{short} is the power dissipation due to voltage switch. The dynamic power dissipation is by far the most significant factor in CMOS model and for this reason, the other ones are ignored in this study:

$$P = KV^2 f \propto Kf^3 \tag{2.16}$$

where the supply voltage can be directly characterized as a linear function of frequency.

By definition, the amount of change in power consumption is equal to the amount

of change in energy consumption per unit time. On this basis, the executional energy required for a workflow is the linear integration of the energy required for execution of each task:

$$E^{execution} = \sum_{t_i \in T} K \times f_{max}^3 \times PT(t_i, r_k) \tag{2.17}$$

where the resources are assumed to work at the highest possible frequency, $f_{max} = 1.0$. The communicational energy required for a workflow is the linear integration of the energy required for each transmission are as follows:

$$E^{transfer} = \sum_{t_i, t_j \in T} K \times f_{min}^3 \times TT(t_i, t_j) \tag{2.18}$$

where the resources are assumed to transmit at the lowest possible frequency, $f_{min} = 0.4$.

To save energy in the cloud system, the frequencies of the resources are set to the lowest level when they are idle. The energy spent by idle resources is calculated with the same way for communicational energy, but the durations of idle times of the resources are considered instead of transfer times. When the resources fail, all their operations stop and no energy is consumed. Ultimately, the computational, communicational, idling and failure energy are considered altogether in the CMOS model for total energy required for a workflow:

$$E = E^{execution} + E^{transfer} + E^{idling} + E^{failure} \tag{2.19}$$

In addition, the energy consumption of other auxiliary operations such as lightning or cooling is not yet taken into the consideration in our study. Since they may be prominent with respect to geographical regions where the cloud data centers are located, they will be handled with in the future work.

2.5. Delays and Uncertainties

Cloud computing is not deterministic environment, where delays and uncertainties may emerge during the execution of the workflows. Two different types of delays and

three different types of uncertainties are considered in our work. The provisioning delay refers to the time needed for virtual machines to be deployed and then booted on physical resources, while the deprovisioning delay refers to the time needed for virtual machines to be released back after the request for shut down. However, deprovisioning delays are generally shorter and more predictable, when compared with provisioning delay. According to the real experimental data [8], the average provisioning and deprovisioning delays for a VM in Linux operating system (OS) in Amazon EC2 [9] are approximately 96.9 and 8.0 seconds, respectively.

On the other hand, the uncertainties include variations in reference task execution times, bandwidth and resource performances. As stated in the application model, the reference execution times of the tasks may vary based on the amount of information the workflow collects at each period. The bandwidth and resource performances may vary based on several factors such as network congestion and instability in supply voltage, respectively. For both delays and uncertainties, their real values during workflow execution are generated by Gaussian distribution as follows:

$$u_i \times N(\mu_i, \sigma_i^2) \tag{2.20}$$

where u_i refers to the baseline delay or uncertainty value, μ_i is its mean value and σ_i^2 is its variance value. The consideration of the indeterminism in cloud computing enables to generate more accurate scheduling solutions.

2.6. Procedure for Workflow Simulation

At the final of this section, the complete procedure for simulation of workflow execution and performance estimation is given in Figure 2.4. The simulation appropriately shows how to provision resources, execute tasks on those resources and estimate the values of objectives for a scheduling solution. If resource r_k is not provisioned in advance in line 3, its lease start time, LST_k, is set to the current time and its lease end time, LET_k, is set to the extension of LST_k by one billing period in line 4. Note that whenever the execution of the task exceeds the lease end time of the resource to which it is assigned, update operation in line 18 is immediately activated in order to prevent the suspension of the ongoing execution. The operation automatically extends the lease end time of

resource by an extra billing period. Then, the values for six given optimization objectives are estimated in line 21 according to the equations (2.6), (2.7), (2.8), (2.10), (2.14) and (2.19).

Input: Set of workflow tasks $t_i \in T$, set of resources $r_k \in R$, task ordering array as *order* and resource mapping array as *map*
Output: Objective Estimations
1: **for** $i = 0$ **to** $|T-1|$ **do**
2: $t_i = order[i]$, $r_k = map[i]$
3: **if** r_k is not leased **then**
4: Set LST_k and LET_k
5: **end if**
6: **if** $task$ is t_{entry} **then**
7: $ST(t_i, r_k) = ready(r_k)$
8: **else**
9: $ST(t_i, r_k) = max(ready(r_k), \max(FT(t_p, r_{k'}))$
10: **end if**
11: $FT(t_i, r_k) = ST(t_i, r_k) + PT(t_i, r_k)$
12: **for** $t_c \in succ(t_i)$ **do**
13: $r_{k'} = map[c]$
14: **if** $r_k \neq r_{k'}$ **then**
15: $FT(t_i, r_k) += TT(t_c, t_i)$
16: **end if**
17: **end for**
18: Update LST_k and LET_k
19: Count N_k
20: **end for**
21: Estimate the objectives with respect to (2.6), (2.7), (2.8), (2.10), (2.14) and (2.19)

Figure 2.4: Pseudocode of Workflow Simulation

3. DYNAMIC MULTI-OBJECTIVE OPTIMIZATION PROBLEM

A multi-objective optimization problem (MOP) is an optimization problem that has two or more objectives which need to be optimized simultaneously, where at least two objectives may be in conflict with one another. Any improvement in one of the conflicting objectives may lead to a deterioration in the other objective. Differently from a single-objective optimization, multi-objective optimization forms multi-dimensional objective space in addition to the decision space. An illustrative example of objective space for a multi-objective optimization problem with two contradictory objectives is presented in Figure 3.1. On the other hand, a dynamic multi-objective optimization problem (DMOP) is a multi-objective optimization problem, in which constraints, objectives, number of objectives or any other problem parameters may dynamically change over time [10]. A dynamic multi-objective optimization problem for minimization of all objectives can be defined as:

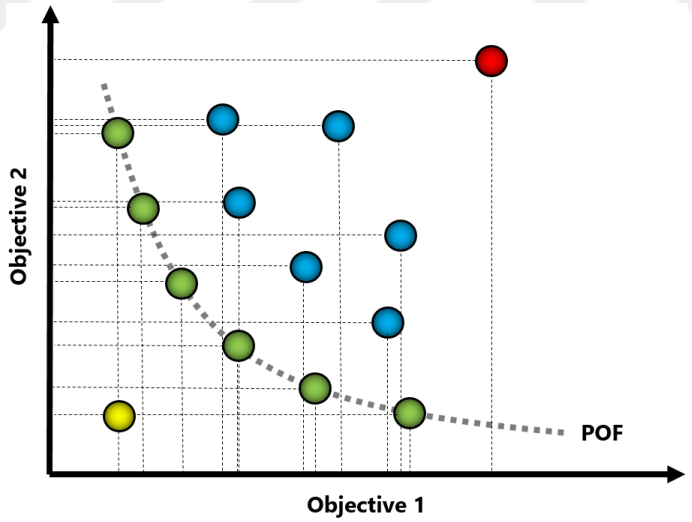


Figure 3.1: An illustration for multi-objective optimization problem

$$\begin{cases} min\ F(x,t) = (F_1(x,t), F_2(x,t), ..., F_o(x,t))^T \\ s.t.\ g_i(x,t) \leq 0, h_i(x,t) = 0 \end{cases} \tag{3.1}$$

where $x \in Z^n : x = (x_1, x_2, ..., x_n)^T$ is the vector of n decision variables in decision space Ω, t is time in time space Ω_t, o is the number of objectives that are considered in the

optimization and F is the set of objectives to be minimized with respect to t [11].

There are different classification techniques for changes in dynamic multi-objective optimization problems in the literature. The most accepted classification is proposed by Farina et al. [12]. According to the classification, changes may happen in decision space or objective space or both of them at the same time:

- Type 1: Only POS changes, but POF remains static.
- Type 2: Both POS and POF change.
- Type 3: Only POF changes, but POS remains static.
- Type 4: Neither POF nor POS changes.

In contrast to single-objective optimization, the presence of conflicting objectives in multi-objective optimization requires a more complex scheme to order and compare the solutions in multi-dimensional objective space. One of the most common scheme for this purpose in the literature is $Pareto-domination$, which is used to test and identify the optimality of a set of solutions to the problem by pair-wise comparison. In a given set, a solution (aka decision vector), x_1, is said to dominate another solution x_2 if the solution x_1 is no worse than the solution x_2 with respect to all objectives values in the objective space and the solution x_1 is strictly better than the solution x_2 in at least one objective. Formally, it is said that x_1 dominates x_2 ($x_1 \prec x_2$) if and only if:

$$\forall i F_i(x_1) \leq F_i(x_2) \wedge \exists j F_j(x_1) < F_j(x_2) \tag{3.2}$$

where $i, j = 1, ..., o$. If a solution, x_1, that are not dominated by any other solutions in a given set, this non-dominated solution is called as *Pareto-optimal* ($\nexists x_2 \in X : \ x_2 \prec x_1$). The set of non-dominated solutions is called as *Pareto-optimal set* ($POS = \{x_1 \in X \mid \nexists x_2 \in X : \ x_2 \prec x_1\}$) in decision space and *Pareto-optimal front* ($POF = \{y = F(x) = (f_1(x), f_2(x), ..., f_m(x))^T \mid x \in POS\}$) in objective space. *True POF* is the maximum frontier of the problem, where the ground truth $Pareto-optimal$ solutions are laid and they cannot be further improvable. As in most non-trivial real-world dynamic multi-objective optimization problems, *True POF* is not known in advance for dynamic workflow scheduling problem in cloud computing. The only solution that would dominate *POF* is called as *utopia* (*ideal*) point which is formed by the best possible values of all

objectives. However, it is impossible to realistically achieve *utopia* point. On the other hand, the opposite point of *utopia* point is *nadir* point which is formed by the worst possible values of all objectives. The *utopia* and *nadir* points are shown with yellow and red, respectively in the Figure 3.1.

3.1. Dynamic Multi-Objective Optimization Algorithms

There are various dynamic multi-objective optimization algorithms to resolve the dynamic multi-objective optimization problems in the current literature. These approaches are aimed to address and cope with the issue of convergence in the optimization environments [13]. They are classified under five different categories: diversity introduction, diversity maintenance, multiple population, memory-based and prediction-based as shown in the Figure 3.2.

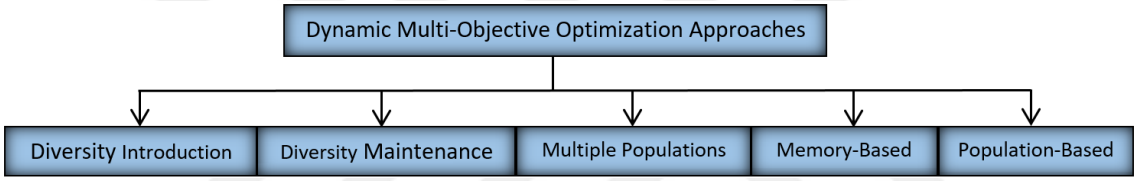


Figure 3.2: The classification of dynamic multi-objective optimization algorithms

- **Diversity introduction**: Convergence to optimal solution is one of the requirements of dynamic multi-objective optimization problems. However, convergence may pose serious challenges since the solutions converged to one region may not detect the change easily or they may not find optimal solutions after change due to lack of diversity. The re-initialization of all population after changes to introduce diversity is the most straightforward way, which may not be appropriate due to loss of all historical information in the most cases. The other way is Hyper-mutation [14], which drastically increases mutation rate after changes. One of the major drawbacks of this approach is that it depends on the detection of changes, which may not be so easy in dynamic optimization environments with less detectable changes.
- **Diversity maintenance**: This approach preserves certain level of diversity during the optimization process instead of sudden diversity introduction. One of the way for this category is Random Immigrants [15], which injects certain number of solutions to the population in each generation. Diversity maintenance is effective especially when the changes are severe. However, continuous preservation of

diversity may slow down the optimization process when the changes are relatively small.

- **Multiple population**: This approach has two or more sub-populations at the same time, where they have distinct tasks to perform to handle with dynamic environments. These tasks may include the detection of changes or convergence to different regions in the search space. Self-Organizing Scouts (SOS) [10] and Adaptive Multi-population Swarm Optimizer (AMSO) [16] can be shown as examples for that technique. Although there are other several techniques use multiple populations, it has problematic issues such as performance due to high computational operations between sub-populations and the prevention of sub-population overlapping.
- **Memory-based**: It uses a implicit or explicit memory scheme to store information about useful solutions and utilize it in later stages. The memory, which is separated from the main population is updated at each fixed time interval during optimization process. Whenever there is a change, the solutions in the memory are evaluated and the best ones are inserted to the population. One of the representative algorithm for this approach is Memory / Search algorithm [17]. Memory-based approach is efficient for problems, where changes are periodic. The redundancy of information and lack of periodicity in changes are still the problems to be solved for this approach.
- **Prediction-based**: This approach aims to exploit information about the solutions and environments of the past optimization environments to estimate the future behaviour and potential optimal solutions in the next environments. There are many prediction-based papers in the literature [18, 19], which incorporate into different techniques such as auto-regression, neural network and support vector machine. The approach works efficiently in dynamic environments, where changes have predictable characteristics. On the other hand, the lack of enough training data or the duration of training process are still important factors to take into the consideration for this approach.

4. DYNAMIC WORKFLOW SCHEDULING PROBLEM

In this section, the details of the dynamic version of the workflow scheduling problem are presented. Those details include both the types of dynamisms in the problem which are resource failures and changing number of objectives; and the effects of the dynamisms on the dynamic multi-objective optimization algorithms. In addition, the state-of-art algorithms that are adapted from the literature are presented, which is our first solution to the problem. A novel prediction-based dynamic multi-objective optimization algorithm incorporated with neural network (NN-DNSGA-II) is also presented in this section, which is our second solution to the problem.

4.1. Problem Definition

Workflow scheduling problem becomes more challenging and more realistic when it is addressed in a dynamic environment. In our study, the dynamic environment specifically forms two different types of dynamism. The first type of dynamism for dynamic workflow scheduling is the failures of resources. Especially, the large-scale systems including cloud systems are very prone to the failures, which may have significant impacts on workflow executions. They may even crash a part or whole of the system in the extreme conditions. In case of the failures, the set of resources that are available may vary, which lead POS, POF or both of them to change as well. For instance, the feasible regions may not be feasible any more in *POS* after change. The target algorithms for dynamic workflow scheduling problem in our study are expected to efficiently track and converge new POF before the next failures. In the experimental study, they are exposed to varying frequency and size of resource failures.

Changing number of objectives is the second type of dynamism considered for workflow scheduling in this study. It may emerge when workflows arrive at the system with different execution requirements. A workflow scheduler of cloud system should be capable of satisfying changing workflow requirements over time. This dynamism specifically alters the shape of objective space of the problem. As shown in the Figure 4.1, an increase in number of objectives leads to the expansion of POF, which deteriorates the diversity of the population [6]. This poses challenges among the solutions in terms of Pareto-domination since it gets harder for a solution to be better than another one in

every objective. This prevents algorithms from converging close to the POF. On the other hand, as shown in the Figure 4.2, a decrease in number of objectives leads to contraction of POF, which creates densely populated similar and duplicate solutions [6]. In order to generate test scenarios for this type of dynamism, the following generalization of the periodic change in objective size is used:

$$o(t) = o(t+c) = \begin{cases} m & t = 0 \\ m+k & t \in [a,b] \\ m-k & t \in [b,c] \end{cases} \tag{4.1}$$

where m the number of objectives at the beginning, k is severity of change or the number of objectives that can change at a time, t is time; and a, b and c show the certain discrete points in time. According to the equation 4.3, the objective size of optimization gradually increases from a to b and then decreases from b to c.

For both dynamic extensions of workflow scheduling, the problem stays stationary between two successive changes. The effects of both resource failures and changing number of objectives are reflected to the problem at the end of each period. Whenever the problem changes, the old instance of periodic workflow is considered as completed and dismissed from the system. It is replaced by new instance of the same periodic workflow with the new workloads in the new period.

4.2. Related Work

Workflow scheduling on distributed systems is an extensively studied problem. Workflow scheduling as an optimization problem involves with either single or multiple optimization objectives. Although there are a few workflow scheduling problems that are classified as single objective optimization in the literature [20, 21], most of the existing research have multi-objective characteristics and they are classfied as multi-objective optimization [22, 23, 24, 25]. The main benefit of multi-objective approach is to find a set of trade-off solutions with respect to the objectives considered and leave the decision-making to end users. Apart from those two approaches, the certain works in the literature linearly merges multiple objectives under unified objective with weight values [26, 27, 28]. However, this may not be appropriate to the nature of the real-world

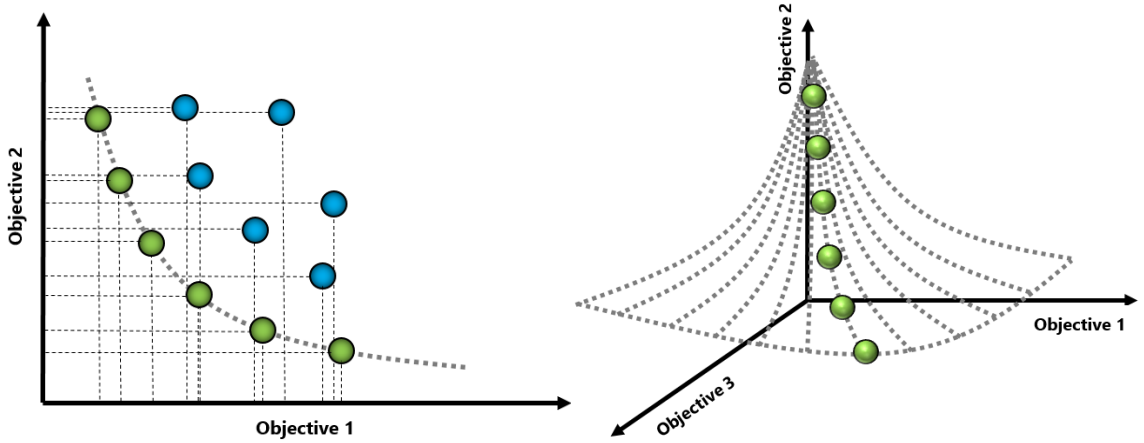


Figure 4.1: An illustration for expansion of POF from 2D to 3D

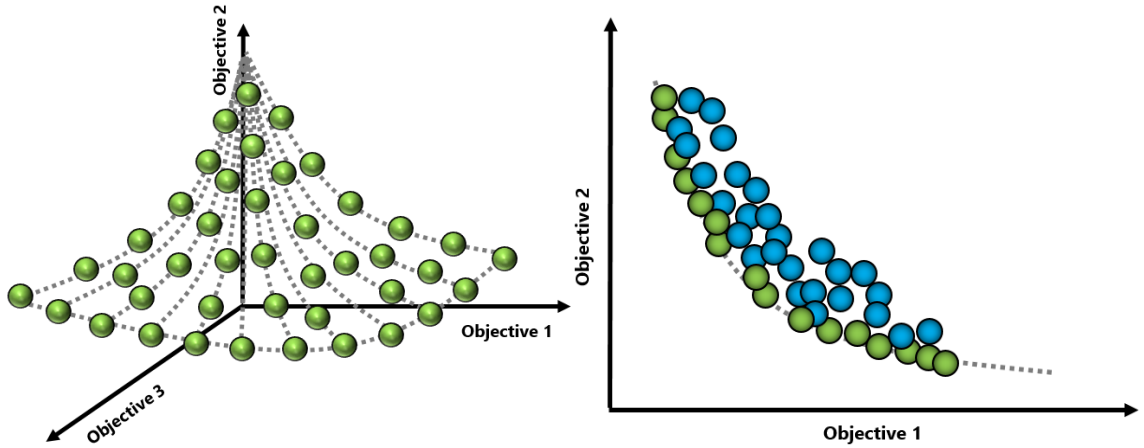


Figure 4.2: An illustration for contraction of POF from 3D to 2D

problem at some conditions such as the existence of partial linear dependency between objectives that are merged or infeasibility of correct determination of weight values of the objectives. Additionally, a linear combination of objectives may not be able to converge to the optimal solutions that are at the concave parts of the Pareto-optimal front of the problem. For workflow scheduling problem in cloud computing, the most common objectives in the literature include makespan [1, 28, 29, 30], cost [1, 20, 21, 31], reliability [22, 27, 30, 31], energy [22, 29, 32], utilization [28, 33, 34, 35] and degree of imbalance [23].

Dynamic workflow scheduling refers to a set of special kind of scheduling techniques that have basic conceptual differences in their definitions in the literature. For a type of dynamic workflow scheduling, the prices of resources may vary over time with respect to the supply and demand of the market. For another type of dynamic workflow

scheduling, information for the tasks in the workflows including their arrivals, workloads and characteristics are unknown in advance and they keep constantly changing during the optimization. On the other hand, dynamic workflow scheduling may also refer to some problems where scheduling decisions are made at runtime [36, 37, 38]. In this study, the definition of dynamic workflow scheduling is extended by considering failures of resources and changing number of objectives due to changing workflow requirements.

Regardless of the efforts in the literature to model and improve the reliability of cloud computing, the failures of resources are still challenging issue to resolve for scheduling algorithms. To understand the characteristics of the failures, Iosup et al. [7] assesses the resource availability data in large systems based on long-term traces and provides sufficient statistical background for frequency, severity and duration of the failures. To enhance the reliability against those failures, fault-tolerant scheduling is mainly considered as effective in the literature, which prevents wastage of money, time and energy in cloud computing to a large extent. Fault-tolerant scheduling techniques mainly include task rescheduling, task replication and check-pointing [39, 40, 41, 42]. For instance, Plankensteiner et al. [41] uses task rescheduling technique with the proposed resubmission impact while meeting deadline constraint in clouds.

The dynamically changing number of objectives in a dynamic multi-objective optimization has been rarely considered so far in the literature [6, 43, 44, 45, 46, 47]. For the first time, Chen et al. [6] systematically analyses the potential effects of dynamically changing number of objectives both on the frontier of the problems and on the performance of the algorithms for various real-life scenarios such as software development life cycle, project scheduling and heterogeneous multi-core systems. It especially emphasizes the importance of changing number of objectives for clouds, since more variety of real-life scheduling scenarios with dynamically changing objectives may appear in clouds [6]. Chen et al. [46] makes some attempts to apply it to scheduling problems in cloud computing. However, it remains rather limited since the cases involving more than two objectives are not investigated. In addition, the main motivation for dynamically changing objectives in that paper is to find at least one feasible solution in the environment, where no feasible solutions previously exist. However, it does not consider changing number of objectives any more after it finds feasible solutions.

Cloud computing in general is a stochastic environment, which may have sources for different types of delays and uncertainties. The major sources of delays include provisioning and deprovisioning delays in acquiring and releasing virtual machines (VMs) while the major sources of uncertainties include variations in task runtimes, resource performance and network performance. However, there are very few papers that address the delays and uncertainties in the literature and examine their effects on the algorithms proposed for cloud computing [34, 35, 48]. If delays and uncertainties are not considered in scheduling, their cumulative propagation may results in negative impacts on the efficiency of scheduling solutions, which may cause discrepancy between the expected and actual results of the solutions. Therefore, the scheduling algorithms addressed for cloud computing have to be aware of those unexpected conditions and generate more robust solutions against them.

The existing dynamic multi-objective evolutionary techniques can be classified under several categories with respect to the ways they cope with the dynamism. Diversity introduction [49, 14, 50], diversity maintenance [45, 15], multi-population [51, 52], memory-based [53, 54, 55] and prediction-based [56, 57, 18, 19, 58, 59] are among the major categories in the literature. The prediction-based models has especially drawn much attention in parallel with increasing popularity of machine learning techniques. The prediction-based models in this domain can be used for forecasting the time for environmental change or tracking the moving optimal solutions over time. Liu et al. [58] adapted neural network to predict the change direction of optimal solutions to track them in dynamic environment for a single objective optimization problem, where useful knowledge is extracted from past experiences in previous environments to make predictions in the future environments. On the other hand, Jiang et al. [59] adapted support vector machine (SVM) to NSGA-II algorithm for the dynamic multi-objective optimization problem, where the solutions in the past Pareto-optimal fronts are labelled as positive while the rest of the solutions are labelled as negative examples.

The main distinction of our work from the other works in the literature is that the dynamic workflow scheduling problem in cloud computing is modelled as dynamic multi-objective optimization problem by incorporating two types of dynamism which are failures of resources and changing number of objectives. A novel

prediction-based dynamic multi-objective evolutionary algorithm (NN-DNSGA-II) is proposed. Additionally, five existing multi-objective evolutionary algorithms are adapted from the literature. They optimize up to six different objectives including minimization of makespan, cost, energy and degree of imbalance; and maximization of reliability and utilization. The algorithms also take several cloud-specific delays and uncertainties into the account while scheduling the workflows.

4.3. Proposed Solutions

4.3.1. Adaptation of Multi-Objective Evolutionary Algorithms

As part of this work, well-known multi-objective evolutionary algorithms are adapted from the literature for solving dynamic workflow scheduling problem in cloud computing. Specifically, four dynamic variants of the Non-dominated Sorting Genetic Algorithm II (the DNSGA-II-A [49], the DNSGA-II-B [49], the DNSGA-II-HM [14], the DNSGA-II-RI [15] algorithms) and one dynamic variant of the MOPSO algorithm (DMOPSO [60]) are considered. According to the classification given in this section, while the DNSGA-II-A, the DNSGA-II-B, the DNSGA-II-HM and the DMOPSO algorithms are diversity introduction based approaches, the DNSGA-II-RI algorithm is a diversity maintenance based approach and finally the proposed NN-DNSGA-II algorithm is a prediction-based approach. In this section, their details and provide their pseudocodes are briefly summarized.

- DNSGA-II-A [49]: replaces certain number of solutions in the population with randomly generated solutions after changes in order to increase diversity among population. The solutions to be replaced are randomly selected from the population with respect to parameter of population replacement rate, θ_{prr}. The total number of solutions for the replacement is calculated by $\theta_{prr} \times |P|$, where $|P|$ represents the size of the population. The main motivation of the algorithm is to relocate a part of the solutions in order to explore relatively distant regions in the search space so that they can track POS of the new optimization environment. This algorithm may especially perform better in the problems, where objectives, constraints or other parameters are exposed to large changes. A sample pseudocode for the DNSGA-II-A algorithm is presented in the Figure 4.3, where the steps (lines 7-11)

belong to classical NSGA-II algorithm. Whenever there is a change (line 3), the DNSGA-II-A algorithm replaces certain solutions for response (line 4) and repair infeasible solutions in order to make them feasible again (line 5).

Input: P : Parent population, Q : Child population, R : Merged population, θ_{prr} : Population replacement rate
Output: POS : Pareto-optimal Set, POF : Pareto-optimal Front

```
1:  InitializePopulations(P, Q)
2:  while termination criteria not satisfied do
3:      if change happened then
4:          RelocateIndividuals(P, θ_prr)
5:          RepairIndividuals(P)
6:      end if
7:      R = P ∪ Q
8:      R_1 = FastNonDominatedSort(R)
9:      R_2 = CrowdingDistanceSort(R)
10:     P = SelectBestIndividuals(R, R_1, R_2)
11:     Q = GenerateChildren(P)
12: end while
```

Figure 4.3: Pseudocode of the DNSGA-II-A Algorithm.

- DNSGA-II-B [49]: replaces a percentage of solutions in the population with the mutated versions of the existing solutions instead of randomly generated solutions. The motivation of the algorithm is to explore relatively the region in the vicinity of the existing solutions in the search space. Therefore, this algorithm may perform better in the problems, where objectives, constraints or other parameters undergo small changes. While completely context-free solutions are added to the population in the DNSGA-II-A algorithm, contextual relationship among solutions is preserved in the DNSGA-II-B algorithm. A sample pseudocode for the DNSGA-II-B algorithm is presented in the Figure 4.4.
- DNSGA-II-HM (Hyper-mutation) [14]: re-diversifies the population that has converged to optimal solutions of previous optimization environments by adaptation of the genetic mutation rates. After the detection of change, the DNSGA-II-HM algorithm temporarily increases normal mutation rates, θ_m, of task repositioning and resource rescheduling mutation operators to high mutation rates, θ_{hm}, and then decreases them to their initial values again. Since only the population after the change is exposed to high mutation rates which distributes solutions to the nearby

Input: P : Parent population, Q : Child population, R : Merged population, θ_{prr} : Population replacement rate
Output: POS : Pareto-optimal Set, POF : Pareto-optimal Front

```
1:  InitializePopulations(P, Q)
2:  while termination criteria not satisfied do
3:     if change happened then
4:        MutateIndividuals(P, θ_prr)
5:        RepairIndividuals(P)
6:     end if
7:     R = P ∪ Q
8:     R_1 = FastNonDominatedSort(R)
9:     R_2 = CrowdingDistanceSort(R)
10:    P = SelectBestIndividuals(R, R_1, R_2)
11:    Q = GenerateChildren(P)
12: end while
```

Figure 4.4: Pseudocode of the DNSGA-II-B Algorithm.

of their current positions in the search space, this algorithm may perform better in the problems with small changes. A sample pseudocode for the DNSGA-II-HM algorithm is presented in the Figure 4.5.

Input: P : Parent population, Q : Child population, R : Merged population, θ_m : Normal mutation rate, θ_{hm} : High mutation rate
Output: POS : Pareto-optimal Set, POF : Pareto-optimal Front

```
1:  InitializePopulations(P, Q)
2:  while termination criteria not satisfied do
3:     if change happened then
4:        RepairIndividuals(P)
5:        θ = θ_hm
6:     end if
7:     R = P ∪ Q
8:     R_1 = FastNonDominatedSort(R)
9:     R_2 = CrowdingDistanceSort(R)
10:    P = SelectBestIndividuals(R, R_1, R_2)
11:    Q = GenerateChildren(P)
12:    θ = θ_m
13: end while
```

Figure 4.5: Pseudocode of the DNSGA-II-HM Algorithm.

- DNSGA-II-RI (Random immigrants) [15]: Different from the previous algorithms that increases diversity after the changes in the environment, it aims to maintain

diversity by continuously replacing certain number of solutions from the population with randomly generated new solutions at each individual evolutionary generation. The number of solutions for the replacement in the DNSGA-II-RI algorithm is based on population replacement rate, θ_{prr}. One of the major advantages of the DNSGA-II-RI algorithm over the other algorithms is that it does not need to explicitly detect changes to react. This advantage especially asserts itself in the optimization environment, where relatively large changes occur. A sample pseudocode for the DNSGA-II-RI algorithm is presented in the Figure 4.6.

Input: P : Parent population, Q : Child population, R : Merged population, θ_{prr} : Population Replacement Rate
Output: POS : Pareto-optimal Set, POF : Pareto-optimal Front

```
InitializePopulations(P, Q)
while termination criteria not satisfied do
    if change happened then
        RepairIndividuals(P)
    end if
    RelocateIndividuals(P, θ_prr)
    R = P ∪ Q
    R_1 = FastNonDominatedSort(R)
    R_2 = CrowdingDistanceSort(R)
    P = SelectBestIndividuals(R, R_1, R_2)
    Q = GenerateChildren(P)
end while
```

Figure 4.6: Pseudocode of the DNSGA-II-RI Algorithm.

- DMOPSO [60]: It is a dynamic and multi-objective extension of the particle swarm optimization. As all previous algorithms are based genetic algorithm, the DMOPSO algorithm is the only algorithm which is based on particle swarm optimization in our study. The particle swarm optimization is adapted for multi-objective optimization problems by incorporating with $Pareto-domination$ mechanism. Different from its other adaptations, MOPSO runs an external repository scheme, which is similar to archive scheme in NSGA-II in order to record the optimal solutions in the past optimization environments. While the particles in classical PSO use positions of best solutions in their neighbourhoods to update their velocities and positions, the particles in MOPSO use positions of single global optimal solution in the repository along with their current velocities and their personal best positions. This global

optimal solution is found by adaptive grid mechanism. MOPSO incorporates into a special mutation operator, which gradually decreases mutation rate with respect to the number of generations. This enables exploration capability in early stages and exploitation capability in late stages of the optimization process. The MOPSO algorithm is adapted for dynamic multi-objective optimization problem by using random re-initialization of certain number of solutions in the population when a change happens, as used in the DNSGA-II-A algorithm. A sample pseudocode for DMOPSO is presented in Figure 4.7.

Input: S : Swarm, P: Repository Particle, R: Repository, θ_{prr} : Population Replacement Rate
Output: POS : Pareto-optimal Set, POF : Pareto-optimal Front

```
1:  InitializeSwarm(S)
2:  while termination criteria not satisfied do
3:      if change happened then
4:          RelocateParticles(S, θ_prr)
5:          RepairParticles(P)
6:      end if
7:      P = SelectRepositoryParticle(R)
8:      S = UpdateSwarm(S, P)
9:      S_1 = FastNonDominatedSort(S)
10:     R = UpdateRepository(S, S_1)
11: end while
```

Figure 4.7: Pseudocode of the DMOPSO Algorithm.

4.3.2. A New Prediction-Based Dynamic Multi-Objective Evolutionary Algorithm (NN-DNSGA-II Algorithm)

The dynamic workflow scheduling problem is a multi-objective optimization problem with dynamic characteristics. It expects algorithms to react to changes effectively and track the changing POF over time as close as possible. Not only they should converge to new optimal solutions for a time span that is allowed before the next change, but also they should avoid the premature convergence at the same time. These requirements disclose why leading static multi-objective algorithms in the literature are not appropriate for dynamic multi-objective optimization problems. In this work, a novel prediction-based dynamic multi-objective evolutionary algorithm (NN-DNSGA-II)

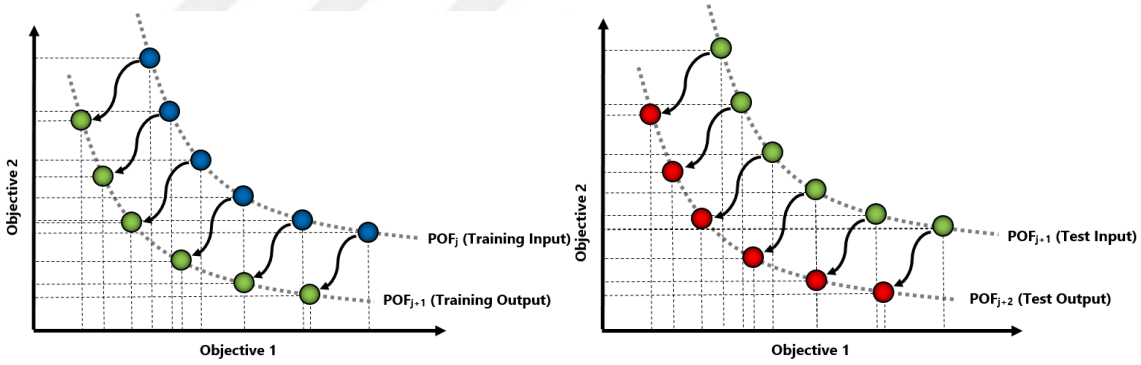
is proposed. Our algorithm integrates neural network with NSGA-II algorithm [61], which is one of the most efficient algorithm in solving static multi-objective optimization problems.

Although there are increasing number of dynamic multi-objective evolutionary algorithms (DMOEAs) in different categories proposed to solve dynamic multi-objective optimization problems in the literature, prediction-based methods mostly attract the attention of the researchers in parallel with recent advances in machine learning. The prediction-based methods mainly integrate dynamic multi-objective evolutionary algorithms with selected well-known machine learning techniques seamlessly [56, 57, 18, 19, 58, 59]. The corresponding strategy that is achieved by the integration in our work targets to solve dynamic workflow scheduling problem by gathering the history of the past POS information and exploit it to train the neural network to estimate next POS at the problem. It first aims to reveal the predictable change patterns behind the displacements of POS in two successive scheduling periods. Then, it utilizes the newly generated solutions at the end of the estimation as initialization points in the search space for the next scheduling period.

The correlative relations are involved within the different components of the dynamic workflow scheduling problem, which can be utilized for the efficient resolution of problem. The first correlation is among the pairs of successive optimization environments and the second correlation is among the pairs of tasks and resources assignment. The regions that are affected by the changes are relatively small portion of the search space. The rest of the regions that are relatively large remain rather unchanged, which constitutes a strong positive correlation between the optimization environments before and after the changes. The trade-offs between conflicting objectives in our multi-objective optimization problem result in correlation between the pairs of tasks and resources for the optimal assignments. In this section, it is aimed (1) to establish connections between optimal solutions in two successive scheduling periods, (2) to reveal dynamics patterns behind the movements of optimal solutions in the search space and finally (3) to estimate the new locations of Pareto-optimal solutions as accurate as possible with the help of neural network.

As can be seen in Figure 4.10, NN-DNSGA-II starts with random re-initialization

of parent and child populations with N number of individuals for each (line 1). When a change happens in the environment (lines 3-10), the multi-layer feedforward neural network is activated. The nearest solutions between POF_j and POF_{j+1} of two successive environments are paired at first (line 4) and then these pairs are included to the training set (line 5). The Pareto-optimal solutions from the first of the successive environments are treated as input data and the Pareto-optimal solutions from the second of the successive environments are treated as output data. Each input solution can corresponds to only one output solution, which means there is one-to-one relationship between input and output sets. Each solution is only allowed to be paired once and therefore the number of paired solutions is equal to the minimum of number of input or output solutions. The pairing process can be seen in the Figure 4.8, where the solutions (blue) in POF_j of the second last environment and the solutions (green) in POF_{j+1} of the last environment are paired as input and output. Then, they are added to training set to generate new potential optimal solutions (red) in POF_{j+2} of the the next environment based on the solutions (green) in POF_{j+1} of the last environment.



(a) Training Phase **(b)** Testing Phase

Figure 4.8: The optimal solutions pairs between previous consecutive $POFs$ in training phase to generate potential optimal solutions for the next POF in testing phase

The representation of the movements of optimal solutions in successive optimization environments needs a pairing process. However, pairing of the closest solutions in terms of Euclidean distance in decision space has an $O(n^2)$ time complexity, where n is the number of tasks in the workflow. This complexity leads to very high running times, especially when scheduling large workflows that have 1000 tasks or more. Instead, the pairing of the closest solutions in terms of Euclidean distance in objective space is

used. The distance between two solutions in the objective space is found as follows:

$$\sqrt{\sum_{i \in o} (F_i(x) - F_i(y))^2} \tag{4.2}$$

where o is the number of objectives considered, x and y are two solution vectors in objective space, $F_i(x)$ and $F_i(y)$ are the objective values of solution vectors in the objective i. The pairing in the objective space is insensitive to the number of tasks. It is only interested in the number of objectives, which is relatively very small compared with number of tasks. A pseudocode of pairing process is presented in the Figure 4.9.

Input: POF_j: POF of the second last environment, POF_{j+1}: POF of the last environment, POF_{j+2}: POF of the next environment, D: Distance matrix
Output: T: Training Set

```
1:  for S ∈ POF_j do
2:      for T ∈ POF_{j+1} do
3:          D[S,T] = CalculateDistance(S,T)
4:      end for
5:  end for
6:  while POF_j ≠ ∅ and POF_{j+1} ≠ ∅ do
7:      [S,T] = SelectNearestSolutionPair(D)
8:      T = T ∪ [S,T]
9:      POF_j = POF_j - {S}
10:     POF_{j+1} = POF_{j+1} - {T}
11: end while
```

Figure 4.9: Pseudocode of Pairing Process.

The neural network goes through the training phase by using the cumulative training set until the given number of epoch is satisfied (line 6). Then, it becomes ready to predict the future locations of Pareto-optimal solutions. In the prediction process, the most recently found Pareto-optimal solutions is fed to the network as input to generate output solutions for the initial population of the next environment (line 7). The new solutions are replaced with the randomly chosen solutions from the parent population (line 8). These solutions are used to speed up the convergence of the evolutionary algorithm in new optimization environment. At the end, the infeasible solutions are repaired to make them feasible again (line 9).

Input: P : Parent population, Q : Child population, R : Merged population, T : Training Set, N : Neural network-generated solutions
Output: POF : Pareto-optimal Front

```
1:  InitializePopulations(P, Q)
2:  while termination condition not satisfied do
3:    if change happened then
4:      [T_j.Input, T_{j+1}.Output] = Pair(POF_j, POF_{j+1})
5:      T = T ∪ [T_j.Input, T_{j+1}.Output]
6:      TrainNeuralNetwork(T)
7:      N = TestNeuralNetwork(T_{j+1}.Output)
8:      P = ReplaceIndividuals(P, N)
9:      RepairIndividuals(P)
10:   end if
11:   R = P ∪ Q
12:   R_1 = FastNonDominatedSort(R)
13:   R_2 = CrowdingDistanceSort(R)
14:   P = SelectBestIndividuals(R, R_1, R_2)
15:   Q = GenerateChildren(P)
16: end while
```

Figure 4.10: Pseudocode of the NN-DNSGA-II Algorithm.

Our neural network model that is integrated into the optimization environment has three-layered architecture as shown in Figure 4.11: an input, a hidden and an output layer. The number of neurons in the input and the output layers is equal to the number of tasks in the workflow; and the number of neurons in the hidden layer is equal to twofold of the number of tasks. The neurons in the input layer take the resource numbers as input, which range from zero to maximum number of resources in the system and correspondingly, the neurons in the output layer generate the resource numbers as output in the same range. Mean Absolute Error (MAE) is used as linear loss function to measure the proximity between predicted and actual results. In addition, the standard back propagation method is used and the maximum number of epoch is set to ten for training phase in our neural network model. Finally, the current system needs further development in order to handle with (1) small training data in the first few changes, (2) relatively higher running times and (3) and full parameter study for better performance.

The steps (lines 11-15) in Algorithm 2 are the standard run of NSGA-II, where it first merges the parent and child population into single population with $2N$ individuals

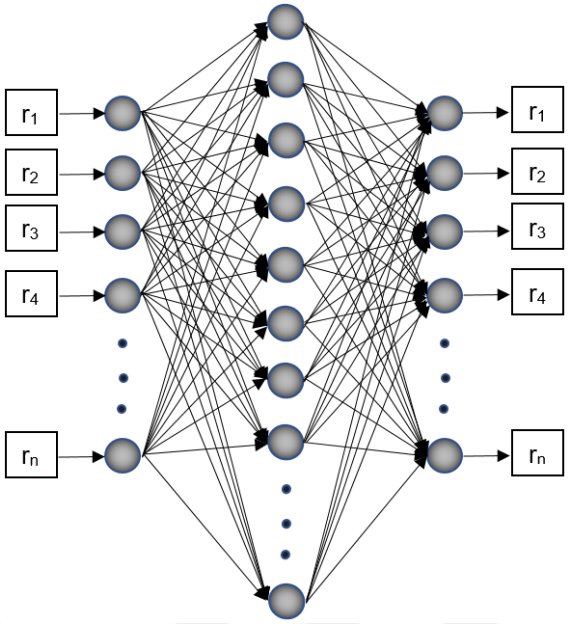


Figure 4.11: The example model of our neural network system

(line 11). This operation ensures the elitism in the environment, which means that the best solutions in terms of each objective are protected from drifting away in evolutionary cycles. The individuals altogether are sorted into different non-domination and crowding density levels with respect to fast non-dominated sorting and crowding distance sorting scheme, respectively (lines 12-13). From $2N$ individuals, the best N ones are selected for parent population of the next generation (line 14). In selection process, the individuals with lower rank are preferred. If they are at the same rank, the ones from less crowded regions are preferred. At the end, the new parent population are used to generate offspring (line 15).

It should be noted that for the NN-DNSGA-II algorithm and the five given algorithms above, two-dimensional and globally constructed solutions are used. A random initialisation is performed on the solutions, whose lengths are specified by number of tasks in the workflow. The first dimension stores the relative execution order of the tasks and the second one maps the tasks to the resources. The single-point crossover is used as suggested in [62, 63], since it ensures that child solutions are always feasible and do not violate the precedence constraints between tasks in the workflows if both parent solutions are feasible as well. For a randomly selected task, a mutation operator randomly repositions it and another one assign a random resource to it.

4.4. Time Complexity

The time complexity of the algorithms used through the thesis theoretically analysed. For a given DAG with the number of n tasks, the single-point crossover operation has an $O(n)$, task repositioning mutation operation has an $O(n)$ and resource rescheduling mutation operation has an $O(1)$ time complexity. On the other hand, evaluation of each workflow scheduling solution is on the order of $O(n^2)$, which is directly proportional to the maximal number of edges at most between n different tasks. For s different solutions in the population, the complexity of total solution evaluation becomes $O(sn^2)$.

From the viewpoints of dynamic multi-objective optimization algorithms with o different objectives, the highest complexity belongs to the non-dominated sorting procedure, which is on the order of $O(os^2)$ for one evolutionary generation. As a result, the overall complexity of multi-objective evolutionary algorithms for dynamic workflow scheduling in cloud computing with g generations is $O(gsn^2 + gos^2)$. The response operations to dynamism in dynamic variants of the algorithms are performed once environment changes and therefore their complexities are considered minimal and negligible, except from NN-DNSGA-II. Since it consumes relative more time in response operations due to training phase of the neural network.

5. EXPERIMENTAL STUDY

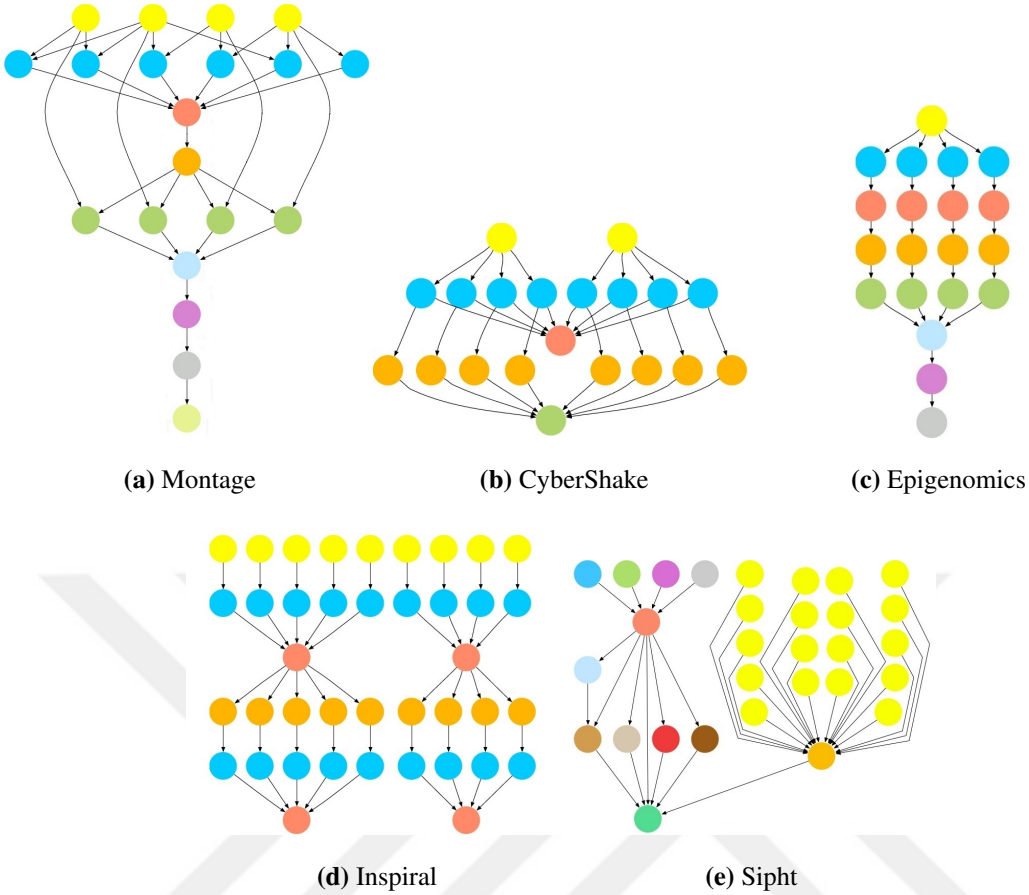
In this chapter, our experimental setup is presented, which is followed by comparison metrics.

5.1. Experimental Setup

5.1.1. Scientific Workflows

Pegasus Workflow Management System [64] is a framework which incorporates into numerous real-world workflow applications. Those applications are mostly used as benchmarks in the literature in order to measure the performances of scheduling algorithms under different scenarios. In our study, the algorithms are evaluated by using ten selected real world workflows with approximately 100 and 1000 tasks from Pegasus Workflow Management System. They include Montage from astronomy, CyberShake from geology, Epigenomics from bioinformatics, Inspiral from astrophysics and Sipht from bioinformatics. The workflows that have less than 100 tasks are not considered in our work since they are not capable of simulating competitive and discriminative environments for the algorithms. The Figure 5.1 symbolically represent their symbolic topological structures such as the task pipelines, data distributions and aggregations. The workflows that are used in this work are briefly described below.

- Montage: It is an astronomy workflow application that is used by NASA/IPAC Infra-red Science Archive. It generates mosaics (composite images) of sky by assembling the set of individual astronomical images. This workflow enables to produce images of multiple regions of sky that are too large to be produced by single cameras. Montage is classified as I/O intensive workflow that requires the efficient transmission of large amount of output data.
- CyberShake: It is geology workflow application that is used by Southern California Earthquake Center. This workflow basically generates synthetic seismogram to characterize the probabilistic earthquake hazards in a region of interest. CyberShake is classified as data and memory intensive workflow that requires both efficient transmission and storage of large amount of output data.
- Epigenomics: It is bioinformatics workflow application is used by USC Epigenome Center in order to map the epigenetic state of human cells. This workflow automates



(a) Montage **(b)** CyberShake **(c)** Epigenomics

(d) Inspiral **(e)** Sipht

Figure 5.1: Scientific workflows from Pegasus Workflow Management System [64]

the execution of various genome sequencing operations such as filtering out noisy and contaminating sequences, generating global mapping of sequences and identifying the density of sequences in the genome. Epigenomics is classified as CPU intensive workflow that requires high processing speed.

- Inspiral: It is astrophysics workflow application in order to detect the gravitational waves that are produced by different celestial events in the universe. LIGO Inspiral, which stands for Laser Interferometer Gravitational Wave Observatory, is used to analyse data of binary systems such as neutron stars or black holes. Inspiral is classified as memory intensive workflow.
- Sipht: It is bioinformatics workflow application is used by Harvard University in order to search for small RNAs which regulates several vital processes of bacteria. This workflow stands for sRNA identification protocol using high-throughput technology. As Epigenomics, Sipht is classified as CPU intensive workflow.

The numerical characteristics of all the workflows used in our study are given in the Table 5.1, which includes the number of tasks, the number of edges, average output data size and average reference execution time of tasks in workflows.

Table 5.1: Characteristics of Workflows [64]

Workflow	Tasks	Edges	Avg. Data Size	Avg. Task Reference Time
Montage 100	100	433	3.23MB	10.58s
Montage 1000	1000	4485	3.21MB	11.36s
CyberShake 100	100	380	849.60MB	31.53s
CyberShake 1000	1000	3988	102.29MB	22.71s
Epigenomics 100	100	322	395.10MB	3954.90s
Epigenomics 1000	997	3228	388.59MB	3858.67s
Inspiral 100	100	319	8.93MB	206.12s
Inspiral 1000	1000	3246	8.90MB	227.25s
Sipht 100	100	335	6.27MB	175.55s
Sipht 1000	1000	3528	5.91MB	179.05s

5.1.2. IaaS Platform

In this work, IaaS platform of Amazon EC2 is adopted, which allows to dynamically resize computational capacity with respect to the needs in terms of on-demand, reserved or spot resources. The IaaS platform offers six on-demand resource types, which have different computational speed and prices. In order to support different requirements of workflows that arrive at the system, the computing units of the resources spans the range from 6.5 to 345 and prices of the resources spans the range from \$0.1 to \$4.608. The technical specifications of the resources are given in Table 5.2, which are compatible with US East (Ohio) region of Amazon EC2 [9].

5.1.3. Other Parameter Settings

The delays for resource provisioning and deprovisioning are generated by Gaussian distributions in seconds, $N(100,10)$ and $N(10,2)$, respectively. They are based on real experimental evidences provided in the work [8]. As in the real-world cases, the deprovisioning delay is more predictable than provisioning delay in our study. On the

Table 5.2: Instance Types and their Specifications [9]

Type	Compute Unit (CU)	Price ($)
m4.large	6.5	0.1
m5.xlarge	16	0.192
m5.2xlarge	31	0.384
m5.4xlarge	60	0.768
m5.12xlarge	173	2.304
m5.24xlarge	345	4.608

other hand, the uncertainties for reference task execution times, resource and bandwidth performance are also generated by multiplying the original values with a random Gaussian number from $N(1.0, 0.1)$.

The dynamic multi-objective optimization of dynamic workflow scheduling in cloud computing is consisted of multiple periods. The number of generations reserved for each period is equal to the division of total number of generations by total number of changes in the environment. For genetic algorithm-based dynamic multi-objective evolutionary techniques, while the crossover rate is 100%, the population replacement rate is 10% and the mutation rates for both task repositioning and resource rescheduling are 1%. Only the DNSGA-II-HM algorithm can increase both mutation rates from normal rates of 1% to high rates of 10% when there is a change. The tournament selection is used for parent selection, where the tournament size is set to 10. The logarithmic time-scale has been employed in resource failures for Weibull distribution [7]. The billing period for resources is 60 minutes and all partial usages are rounded up, which is compatible with Amazon EC2. The bandwidth is 1 Gbps in the system. The utilization and the degree of imbalance are used with the remaining four objectives only in the experiments to test the algorithms under changing number of objectives.

For particle swarm optimization based DMOPSO algorithm, inertia weight is 0.6 and the coefficients for both cognitive and social components are set to 2. The random parameters for both cognitive and social components take value in the range of $[0, 1]$. As suggested in the original paper [60] to balance its exploration and exploitation capability, the mutation rates in DMOPSO are gradually decreased with respect to the number of

iteration as follows:

$$\left(\frac{g_{total} - g_{current}}{g_{total}}\right)^{\frac{5}{\theta}} \tag{5.1}$$

where g_{total} is total number of generations reserved for each period, $g_{current}$ is the current number of generations that have passed and θ is the mutation rate. Finally, Table 5.3 lists the values for all other default parameters.

Table 5.3: Default parameter settings

Parameter	Value
Population Size	50
Task Size	100 / 1000
Resource Size	60
Archive / Repository Size	50
Frequency of Change (η, β)	(13, 12)
Severity of Change (η, β)	(2, 2)
Crossover Rate	100%
Population Replacement Rate	10%
Hyper-mutation Rate	10%
Mutation Rate	1%
Number of Runs	30
Stopping Criterion (Generation)	1000
Number of Objectives (Min. - Max.)	3 - 6

5.2. Performance Metrics

The performance metrics are commonly used to evaluate the performance of dynamic multi-objective optimization algorithms in dynamic multi-objective optimization environments. They are classified as metrics targeted for problems with known or unknown *true POF* in the literature.

In this study, three performance metrics are considered to measure the effectiveness of the algorithms to find good non-dominated solutions in dynamic environments, which

are *number of non-dominated solutions* (*NS*), *Schott's spacing* (*SS*) and *Hypervolume* (*HV*). All those metrics are mostly used for dynamic multi-objective optimization problems with unknown true *POF*, which are appropriate to our problem [52].

- **Number of Non-dominated Solutions (*NS*)** [52]: counts the average of the number of non-dominated solutions found at each generation just before the changes in the environment. It is defined as follows:

$$NS = |POF_{known}| \tag{5.2}$$

  where $|.|$ operator returns the amount of solutions that are parts of the known *POF*. For this metric, it is better for the algorithms to find non-dominated solutions as much as possible. However, this may be misleading in case a new solution dominates and discards the solutions already in the known *POF*. From this viewpoint, it is recommended to evaluate the results of this metric along with the results of the other metrics. Number of non-dominated solutions provides only auxiliary information about the performance of the targeted algorithms in order to assist to arrive certain deductions.

- **Schott's Spacing (*SS*)** [65]: measures how regularly the non-dominated solutions are spread into the known *POF* at each generation just before changes. It considers variance between the pair-wise distances of neighbour solutions. It is defined as follows:

$$SS = \frac{1}{|POF_{known}|}\left[\frac{1}{|POF_{known}|}\sum_{s=1}^{|POF_{known}|}(d_s - \bar{d})\right]^{\frac{1}{2}} \tag{5.3}$$

  where d_s is the Euclidean distance between non-dominated solution s and its nearest neighbour; and $\bar{d}$ is the average distance between solutions. The Euclidean distance between a solution s and its nearest neighbour solution $s^{'}$ in objective space is calculated as follows:

$$d_s = \min_{s' \in POF_{known}} \sqrt{\sum_{i \in o}(F_i(s) - F_i(s'))^2} \tag{5.4}$$

  and average distance between solutions is defined as follows:

$$\bar{d} = \frac{1}{|POF_{known}|} \sum_{s=1}^{|POF_{known}|} d_s \tag{5.5}$$

The metric prefers the algorithms that generate the known *POF*, in which the non-dominated solutions are well-distributed and not concentrated in certain regions of the objective space. Schott's Spacing (*SS*) must be equal to zero in the the most ideal case.

- **Hyper-volume (*HV*)** [66]: This metric measures the amount of space covered by non-dominated solutions found at each generation just before the changes, with respect to the pre-defined reference point (see Figure 5.2). A large HV value implies that the solutions found are closer to *true POF* and further from the reference point. The space covered by a single non-dominated solution s is defined as follows:

$$HV(s) = \{s' \in O : s \prec s'\} \tag{5.6}$$

where O represents the objective space and the total space dominated by union of non-dominated solutions is:

$$HV(POF_{known}) = \bigcup_{s \in POF_{known}} HV(s) \tag{5.7}$$

For this metric, the fitness values of the scheduling solutions are normalized by the upper bounds found so far in the experimental study for all workflows. The upper bound for each objective are mostly determined by the large workflows. However, the normalization by such numerically large values hinders the real improvements of the algorithms. That is why, even very small differences among the algorithms in HV metric actually corresponds the very significant improvements in the objectives. After normalization, the reference point for each objective is set to 1.1, as recommended in the following work [67].

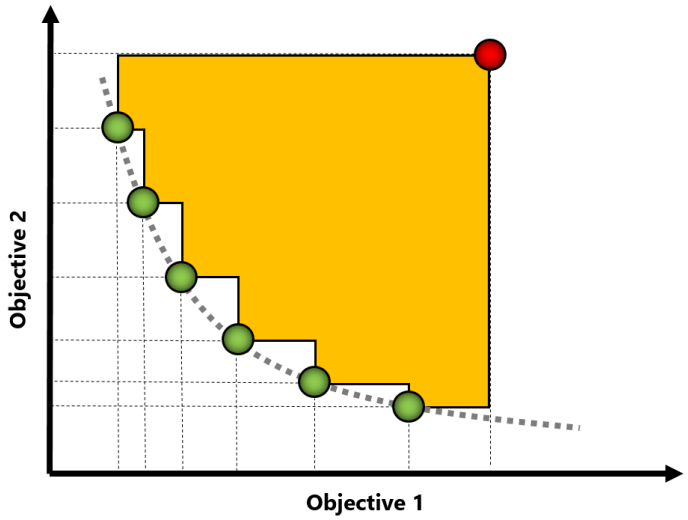


Figure 5.2: The area calculated for Hypervolume metric in two dimensional objective space

6. RESULTS AND DISCUSSION

The results of our experimental evaluation are given in two parts. The first section presents the performance evaluation of the algorithms with respect to resource failures. The second section validates the performance of our algorithms with respect to changing number of objectives.

6.1. Measuring the Effect of Resource Failures

6.1.1. Varying Frequency of Changes

In the first experiment of this category, the effects of varying change frequency are measured on the performance of the algorithms. A scheduling period is assigned to 100 generations in high-frequency cases and 500 generations in low-frequency cases, while the other parameters are set to their default values given in Table 5.3. It is expected to have (1) greater number of non-dominated solutions for *NS*, (2) smaller deviation in distribution of these solutions for *SS* and (3) higher coverage area in objective space for *HV* in low cases, where the algorithms have more time before the next change.

The results for workflows with 100 and 1000 tasks under varying frequency of change are presented in the Table 6.1 and 6.2, respectively. The algorithms tend to find more non-dominated well-distributed and well-converged solutions for most of the cases when $f = 500$ in parallel with our expectations. It can be inferred that the larger and more complex workflows have more impacts on the performances of the algorithms. For instance, compared with other workflows with 100 tasks, Epigenomics 100 is more complex in terms of task size, edge size, average data size or average task reference execution times. For this workflow, the algorithms have notably poor performances in *SS* and *HV* metrics. The performances of algorithms are better when they deals with the workflows with 100 tasks rather than workflows with 1000 tasks.

Among all algorithms, NN-DNSGA-II outperforms the other alternatives, where it provides best *NS* and *HV* values for almost all of the cases. However, there is no algorithm that is superior than the others when *SS* is considered. Overall, NN-DNSGA-II is superior than the other algorithms for 24 out of 30 cases for the workflows with 100 tasks and 19 out of 30 cases for the workflows with 1000 tasks. The superiority of NN-DNSGA-II

is based on its utilization of the predictability of the changes in dynamic workflow scheduling in cloud computing, which results from the positive correlation between pairs of tasks and resources and pairs of two successive optimization environment. On the other hand, DMOPSO is the least appropriate algorithm due to its poor performance in every metrics. The particle swarm optimization was initially proposed for continuous space problems and its performance greatly suffers from the discrete problems and cannot satisfy the requirements of the discrete problems. In order to indicate the significance between the results of the algorithms, the Wilcoxon rank-sum test [68] is carried out at the 0.05 significance level for each table, where the best results are shown as bold. The results with "+" sign in those tables are significantly outperformed by the best results in bold.

Table 6.1: Varying Frequency of Change On Workflows With 100 Tasks

		Montage 100		CyberShake 100		Epigenomics 100		Inspiral 100		Sipht 100	
Algorithm	Metric	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$
DNSGA-II-HM	NS	35.6	44.2+	32.1+	39.9+	42.5+	46.8	44.8	47.4+	47.1+	49.6+
	SS	2.42	0.29	1.88	2.49+	63.66	32.89	1.99	**0.96**	2.63	1.49
	HV	0.9029	0.9038	0.9035	0.9043	0.8595	0.8611	0.9025	0.9042	0.8989	0.9004
DNSGA-II-A	NS	37.8	45.9	34.8+	42.9	44.1	46.3	43.9	46.8+	47.2+	49.5+
	SS	1.88	0.26	2.11	**0.91**	92.88	32.12	1.89	1.04	3.89	3.56
	HV	0.9023	0.9041	0.9012	0.9031	0.8583	0.8616	0.9030	0.9031	0.8934+	0.8967+
DNSGA-II-B	NS	37.9	47.8	33.6+	43.1	41.2+	46.9	45.0	48.5+	46.9+	49.7+
	SS	**0.97**	**0.25**	**1.81**	2.63+	44.59	28.24	**1.09**	1.44	2.52	2.04
	HV	0.9038	0.9039	0.9025	0.9043	0.8550	0.8603	0.9033	0.9045	0.8966	0.8991
DNSGA-II-RI	NS	31.2+	47.5	36.4	40.1+	44.2	46.7	44.2	48.4+	47.6+	48.9+
	SS	2.12	0.67	2.68	1.69	49.72	37.99	2.23	1.14	3.14	2.12
	HV	0.9019	0.9033	0.9012	0.9039	0.8561	0.8585	0.9010	0.9024	0.8966	0.8975+
DMOPSO	NS	16.9+	21.9+	35.3	41.1	23.9+	29.1+	21.6+	25.2+	33.9+	43.9+
	SS	16.6+	14.2+	4.91+	4.75+	400.06+	300.55+	32.52+	18.93+	15.83+	4.29+
	HV	0.8888+	0.8900+	0.8852+	0.8878+	0.7143+	0.7832+	0.8946+	0.8989+	0.8805+	0.8812+
NN-DNSGA-II	NS	**40.3**	**48.1**	**42.3**	**44.1**	**48.2**	**48.8**	**46.6**	**49.9**	**49.9**	**50.0**
	SS	2.62+	1.59+	3.91+	4.31+	**34.1**	**19.12**	5.10+	3.47+	**2.38**	**1.56**
	HV	**0.9089**	**0.9089**	**0.9071**	**0.9074**	**0.8611**	**0.8624**	**0.9054**	**0.9076**	**0.9032**	**0.9065**

6.1.2. Varying Severity of Changes

The performance variations of the algorithms under different severity of changes is investigated in this experiment. The number of resources that fail at every period is 2, 4, 8 out of 60 resources in small, medium and high severity cases, respectively. The severity value for changes refers to the displacement of the optimal solutions. The optimization

Table 6.2: Varying Frequency of Change On Workflows With 1000 Tasks

		Montage 1000		CyberShake 1000		Epigenomics 1000		Inspiral 1000		Sipht 1000	
Algorithm	Metric	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$
DNSGA-II-HM	NS	23.8	**39.8**	48.1	49.8	**43.2**	48.3	38.5	46.1	18.3	21.3
	SS	**2.34**	0.98	1.08	1.13	75.04	63.89	5.25	2.94	3168.02	1690.79
	HV	0.8586	0.8655	0.8560	0.8612	0.075	0.0771	0.8632	0.8763	0.0213	0.0246
DNSGA-II-A	NS	25.8	39.1	48.9	49.7	37.5	48.6	40.4	46.4	16.7	21.2
	SS	3.62	**0.89**	0.87	1.32	201.34	**36.71**	5.66	2.42	3184.73	2209.74
	HV	0.8594	0.8667	0.8552	0.8595	0.0754	0.0772	0.8650	0.8764	0.0218	0.0238
DNSGA-II-B	NS	26.6	37.0	49.4	**49.9**	39.1	48.8	40.6	**47.3**	17.3	20.5
	SS	5.61	1.90	**0.80**	**1.11**	135.86	42.13	**4.70**	**2.25**	2196.05	1783.67
	HV	0.8618	0.8650	0.8560	0.8624	0.0757	0.0773	0.8654	0.8769	0.0219	0.0224
DNSGA-II-RI	NS	26.0	35.3	45.1	49.7	42.0	49.6	40.3	46.8	23.1	28.4
	SS	5.41	4.78	1.10	1.17	100.68	46.11	6.37	3.08	2674.84	**1493.78**
	HV	0.8572	0.8627	0.8538	0.8575	0.0752	0.0772	0.8610	0.8739	0.0210	0.0225
DMOPSO	NS	21.7	25.6	42.2	44.7	14.0	16.3	16.5	21.2	13.6	15.0
	SS	4.71	4.06	2.38	2.51	1362.16	838.46	85.64	52.67	9615.27	9222.60
	HV	0.8489	0.8491	0.8476	0.8472	0.068	0.0690	0.8129	0.8196	0.0163	0.0178
NN-DNSGA-II	NS	**36.8**	**39.8**	**49.8**	45.5	37.9	**49.7**	**40.8**	39.9	**32.6**	**33.6**
	SS	8.78	4.18	8.28	5.60	**60.47**	60.47	104.03	104.87	**2084.53**	2819.11
	HV	**0.9027**	**0.9080**	**0.9005**	**0.8890**	**0.0768**	**0.0775**	**0.8719**	**0.8958**	**0.0322**	**0.0336**

environment that is exposed to high severity of changes moves the optimal solutions away from their current positions, which poses greater challenges for the algorithms in terms of adaptation to the next environment. The recovery and convergence to the optimal solutions phases of the algorithms take longer times. Therefore, it is expected to tend to provide (1) lower number of non-dominated solutions for *NS*, (2) higher distributional deviation between these solutions for *SS* and (3) lower coverage area in objective space for *HV* in high severity cases.

The results for workflows with 100 and 1000 tasks under varying severity of change are presented in the Table 6.3 and 6.4 respectively. It can be inferred from those results that when the landscape of the search space of the problem changes much, the algorithms have difficult times to converge new POF of the next environment. This lack of convergence emerges in the metrics as decrease of *NS* and *HV* values and increase of *SS* values. The algorithms show better performances when the workflows are small or less complex such as Montage 100. The NN-DNSGA-II algorithm outperforms the other algorithms for 35 out of 45 cases for the workflows with 100 tasks; and 32 out of 45 cases for the workflows with 1000 tasks. It mostly outperforms the other algorithms in *NS* and *HV* metrics while it has still some drawbacks to find well-distributed solutions over POF. It

is mainly originated from the fact that NN-DNSGA-II can find non-dominated solutions near the end points of Pareto-front, which results in high distance variations between the extreme solutions. On the other hand, it may be useful to have good solutions from large range of Pareto-front in terms of decision-making. As happened in the experiment to measure the effect of change frequency, DMOPSO is the worst algorithm.

As a part of this experiment, the running times of the algorithms are measured under default parameter settings. The running times of the algorithms in a single period vary over the range of 1.1 to 2.8 seconds for workflows with 100 tasks. On the other hand, their running times vary over the range of 21.1 to 102.9 seconds for the cases with 1000 tasks. The fastest algorithm is DMOPSO in the most cases, but the quality of its scheduling solutions is considerable poor. There are no significant amount of running times between the dynamic extensions of the NSGA-II algorithm. However, the DNSGA-II-RI algorithm relatively finishes later due to its continuous solution injection to the population at every generation. Although the slowest algorithm is NN-DNSGA-II which consumes more time due to initial training phase of the neural network, its running time in the most cases are in the acceptable range.

6.1.3. Varying Resource Size

The performances of the algorithms are also measured on average resource size for various workflows. The average performance of the algorithms in cloud system with 30, 60 and 120 resources is considered for each workflow. The experimental results are presented in the Figure 6.1. The proposed NN-DNSGA-II algorithm apparently outperforms the other algorithms in NS and HV metrics. Its performance in Epigenomics 100 and 1000 workflows is the best one among the workflows since it is the best algorithms with respect to all of the metrics. However, the performance of NN-DNSGA-II in SS metric is not as satisfying as its performance in other metrics for other workflows. In addition, there is a natural degradation in the performances of the algorithms when the workflows are relatively large (i.e., the number of tasks is increased from 100 to 1000).

The algorithms are expected to provision just optimal number of resources with optimal computational speeds in scheduling solutions for higher performances in the given metrics. In case under-provisioning or over-provisioning of resources arises, the

Table 6.3: Varying Severity of Change On Workflows With 100 Tasks

		Montage 100			CyberShake 100			Epigenomics 100		
Algorithm	Metric	$s = 2/60$	$s = 4/60$	$s = 8/60$	$s = 2/60$	$s = 4/60$	$s = 8/60$	$s = 2/60$	$s = 4/60$	$s = 8/60$
DNSGA-II-HM	NS	37.2	35.6	32.9+	36.1+	31.9+	31.9+	45.4+	42.5+	42.1
	SS	1.48	2.42+	1.69	**1.60**	1.88	2.45	40.75	63.66	49.00
	HV	0.9049	0.9029	0.9032	0.9040	0.9035	0.9008	0.8604	0.8595	0.8568
DNSGA-II-A	NS	37.9	37.8	33.2+	36.0+	34.8+	33.0+	44.9+	44.1	43.1
	SS	1.03	1.88	4.42+	2.44	2.11	**2.27**	55.12	92.88	49.13
	HV	0.9045	0.9023	0.9021	0.9039	0.9012	0.9015	0.8619	0.8583	0.8562
DNSGA-II-B	NS	37.9	37.9	33.1+	38.1	33.6+	32.9+	44.1+	41.2+	41.8+
	SS	**0.60**	**0.99**	**1.59**	1.99	**1.85**	2.57	40.45	44.59	48.79
	HV	0.9052	0.9038	0.9011	0.9049	0.9025	0.9023	0.8599	0.8550	0.8542
DNSGA-II-RI	NS	34.0+	31.2+	29.0+	36.5+	36.4	38.2	45.9+	44.2	41.7+
	SS	1.60	2.12	5.62+	3.65+	2.68	2.44	39.13	49.72	87.36
	HV	0.9029	0.9019	0.9017	0.9039	0.9012	0.9011	0.8569	0.8561	0.8532
DMOPSO	NS	15.0+	16.9+	17.2+	33.6+	35.3	31.0+	25.1+	23.9+	19.9+
	SS	16.34+	16.60+	16.21+	5.33+	4.91+	4.43+	399.17+	400.06+	454.54+
	HV	0.8912+	0.8888+	0.8860+	0.8861+	0.8852+	0.8917+	0.7685+	0.7143+	0.4991+
NN-DNSGA-II	NS	**40.5**	**40.3**	**39.0**	**42.9**	**42.3**	**41.0**	**49.9**	**48.2**	**46.7**
	SS	3.45+	2.62+	2.23	4.12+	3.91+	3.03	**31.12**	**34.1**	**41.59**
	HV	**0.9070**	**0.9089**	**0.9062**	**0.9075**	**0.9071**	**0.9070**	**0.8633**	**0.8611**	**0.8554**

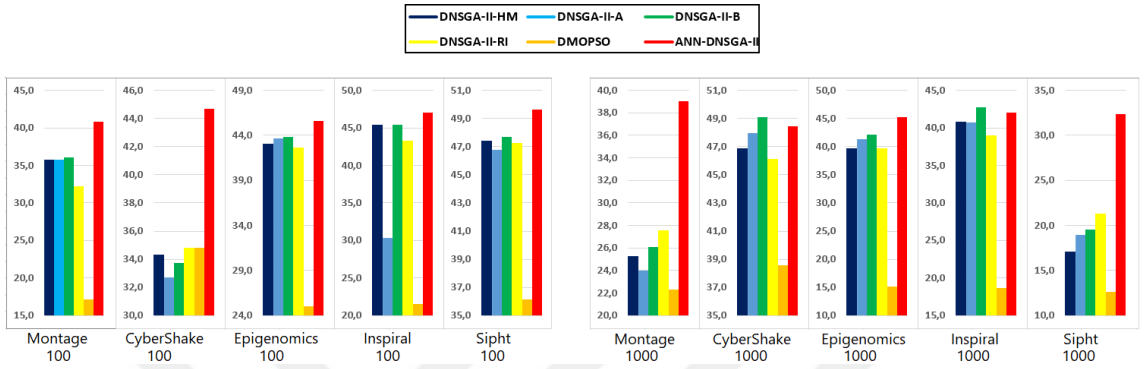
		Inspiral 100			Sipht 100		
Algorithm	Metric	$s = 2/60$	$s = 4/60$	$s = 8/60$	$s = 2/60$	$s = 4/60$	$s = 8/60$
DNSGA-II-HM	NS	46.8	44.8	44.2	47.9+	47.1+	46.9+
	SS	**0.80**	1.99	2.11	2.41	2.63	**2.92**
	HV	0.9035	0.9025	0.9025	0.8996	0.8989	0.8972
DNSGA-II-A	NS	45.1	43.9	43.2	46.9+	47.2+	47.3+
	SS	1.35	1.89	1.70	3.22	3.89	3.07
	HV	0.9039	0.9030	0.9015	0.8980	0.8934+	0.8990
DNSGA-II-B	NS	46.9	45.0	43.2	47.0+	46.9+	46.8+
	SS	1.05	**1.09**	**1.56**	2.97	2.52	2.90
	HV	0.9052	0.9033	0.9033	0.8997	0.8966	0.8960
DNSGA-II-RI	NS	45.7	44.2	41.9	47.8+	47.6+	45.3+
	SS	1.56+	2.23+	3.02+	3.46	3.14	3.22
	HV	0.9044	0.9010	0.9009	0.8985	0.8966	0.8967
DMOPSO	NS	21.9+	21.6+	20.0+	37.1+	33.9+	31.2+
	SS	36.53+	32.52+	26.74+	10.13+	15.83+	11.10+
	HV	0.8967+	0.8946+	0.8930+	0.8900+	0.8805+	0.8779+
NN-DNSGA-II	NS	**47.1**	**46.6**	**44.9**	**49.9**	**49.9**	**49.9**
	SS	4.51+	5.10+	6.13+	**2.14**	**2.38**	4.00
	HV	**0.9077**	**0.9054**	**0.9058**	**0.9060**	**0.9032**	**0.9060**

Table 6.4: Varying Severity of Change On Workflows With 1000 Tasks

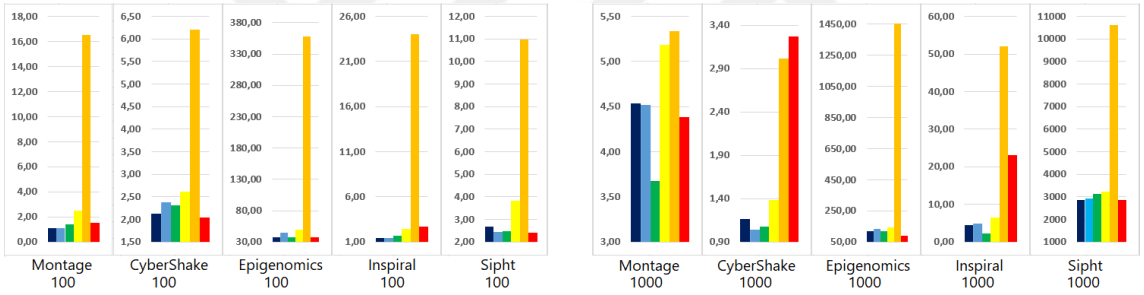
		Montage 1000			CyberShake 1000			Epigenomics 1000		
Algorithm	Metric	$s = 2/60$	$s = 4/60$	$s = 8/60$	$s = 2/60$	$s = 4/60$	$s = 8/60$	$s = 2/60$	$s = 4/60$	$s = 8/60$
DNSGA-II-HM	NS	25.0	23.8	22.1	48.8	48.1	45.9	44.2	**43.2**	36.3
	SS	**2.31**	**2.34**	**2.46**	1.05	1.08	0.9593	61.18	75.04	166.03
	HV	0.8613	0.8586	0.8588	0.8562	0.8560	0.8533	0.0764	0.0750	0.0754
DNSGA-II-A	NS	23.1	25.8	26.4	49.4	48.9	47.0	41.2	37.5	37.5
	SS	3.97	3.62	2.64	**0.97**	0.87	0.82	77.74	201.34	243.98
	HV	0.8635	0.8594	0.8568	0.8569	0.8552	0.8548	0.0760	0.0754	0.0746
DNSGA-II-B	NS	21.3	26.6	26.0	49.5	49.4	**48.1**	44.7	39.1	**37.9**
	SS	3.26	5.61	5.54	1.01	**0.80**	**0.78**	80.97	135.86	353.26
	HV	0.8679	0.8618	0.8570	0.8565	0.8560	0.8538	0.0762	0.757	0.0747
DNSGA-II-RI	NS	26.2	26.0	24.4	48.5	45.1	46.3	43.5	42.0	36.9
	SS	3.19	5.41	5.45	1.01	1.10	0.86	67.65	100.68	171.30
	HV	0.8650	0.8572	0.8576	0.8541	0.8538	0.8532	0.0757	0.0752	0.0746
DMOPSO	NS	21.4	21.7	24.0	41.7	42.2	38.6	14.7	14.0	12.4
	SS	4.21	4.71	6.14	2.39	2.38	3.07	1320.41	13662.16	1355.57
	HV	0.8497	0.8489	0.8420	0.8437	0.8476	0.8406	0.0694	0.068	0.0663
NN-DNSGA-II	NS	**40.3**	**36.8**	**38.3**	**49.9**	**49.8**	45.7	**49.0**	37.9	**37.9**
	SS	8.54	8.78	8.81	8.56	8.28	6.22	**60.44**	**60.47**	**160.37**
	HV	**0.9041**	**0.9027**	**0.8983**	**0.9075**	**0.9005**	**0.8956**	**0.0769**	**0.0768**	**0.0760**

		Inspiral 1000			Sipht 1000		
Algorithm	Metric	$s = 2/60$	$s = 4/60$	$s = 8/60$	$s = 2/60$	$s = 4/60$	$s = 8/60$
DNSGA-II-HM	NS	40.9	38.5	35.2	19.1	18.3	17.2
	SS	3.62	5.25	6.76	2442.60	3168.02	3306.09
	HV	0.870	0.8632	0.8635	0.0222	0.213	0.0222
DNSGA-II-A	NS	44.4	40.4	36.5	17.3	18.3	18.5
	SS	**3.19**	5.66	7.94	3181.69	3168.02	**3073.04**
	HV	0.8696	0.8650	0.8624	0.0221	0.0218	0.0211
DNSGA-II-B	NS	**44.5**	40.6	38.0	17.0	17.3	17.1
	SS	3.99	**4.70**	**6.34**	2110.43	2196.05	3491.11
	HV	0.8690	0.8654	0.8627	0.0222	0.0219	0.0204
DNSGA-II-RI	NS	38.8	40.3	36.36	23.9	23.1	19.8
	SS	5.99	6.37	8.20	2386.81	2674.86	3502.56
	HV	0.8648	0.8610	0.8581	0.0196	0.0210	0.0197
DMOPSO	NS	20.5	16.5	11.5	14.2	13.6	12.2
	SS	47.17	85.64	170.31	9317.39	9615.27	14907.46
	HV	0.8164	0.8476	0.8035	0.0178	0.0163	0.0141
NN-DNSGA-II	NS	39.2	**40.8**	**40.5**	**34.8**	**32.6**	**30.1**
	SS	82.38	104.03	107.58	**2107.00**	**2084.53**	3341.91
	HV	**0.8747**	**0.8719**	**0.8705**	**0.0400**	**0.0322**	**0.0283**

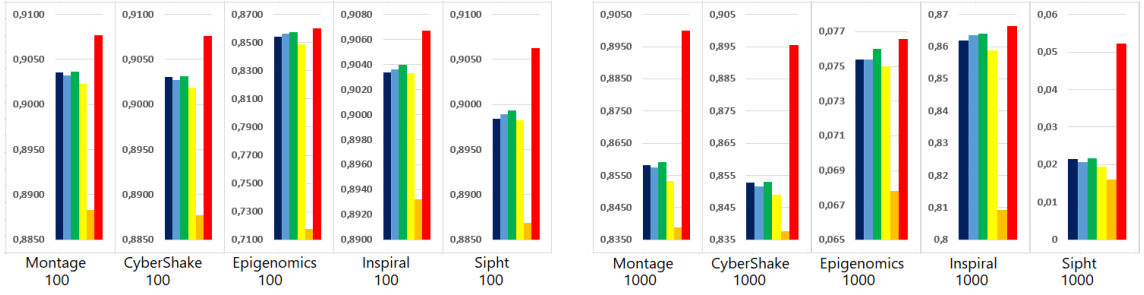
algorithms may lack ability to converge into the optimal solutions in the search space. According to the experimental results of average resource size in overall, it can be inferred that NN-DNSGA-II is the fittest among the algorithms to determine the optimal number of resources with optimal computational powers for dynamic workflow scheduling in cloud computing.



(a) *NS*



(b) *SS*



(c) *HV*

Figure 6.1: Performance of algorithms in (a) *NS*, (b) *SS* and (c) *HV* metrics for average resource size

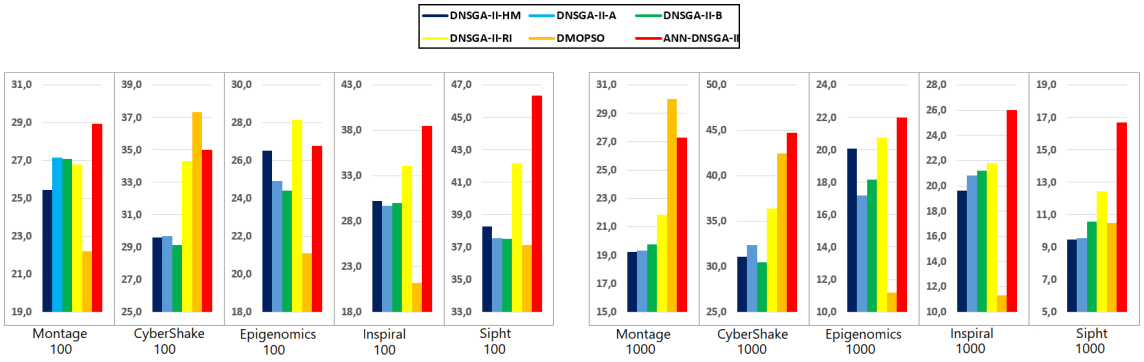
6.1.4. Varying Variance of Uncertainties

The variance of the uncertainties is varied in cloud computing to observe their impacts on the performance of baseline scheduling solutions generated by the targeted algorithms, where the uncertainties include provisioning delay, deprovisioning delay, task execution times, bandwidth and finally resource performance. The higher the variances of the uncertainties are, the more unpredictable the differences between the actual and the estimated finish times of the tasks are. For an uncertainty with estimated base value u_i in the experiment, its actual value is generated by $u_i \times N(\mu, \sigma^2)$. The variance takes the value of 0.1, 0.3 and 0.5 in the test environment with low, medium and high uncertainties, respectively.

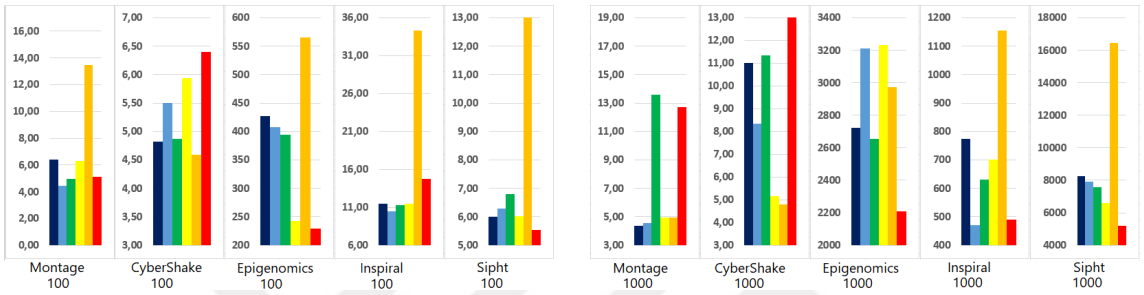
The empirical results for average of varying variance of uncertainties are presented in the Figure 6.2, where the environment with high uncertainties poses greater challenges to the algorithms to cope with. According to the given average results for three levels of environments, the proposed NN-DNSGA-II algorithm outperforms the other algorithms based on *NS* and *HV* metrics for most cases. Especially, its performance in *HV* is notable greater than the other algorithms for the workflows with 1000 tasks. On the other hand, NN-DNSGA-II still suffers from finding sufficiently well-distributed non-dominated solutions except from a few cases including Epigenomics 1000.

6.2. Measuring the Effect of Changing Number of Objectives

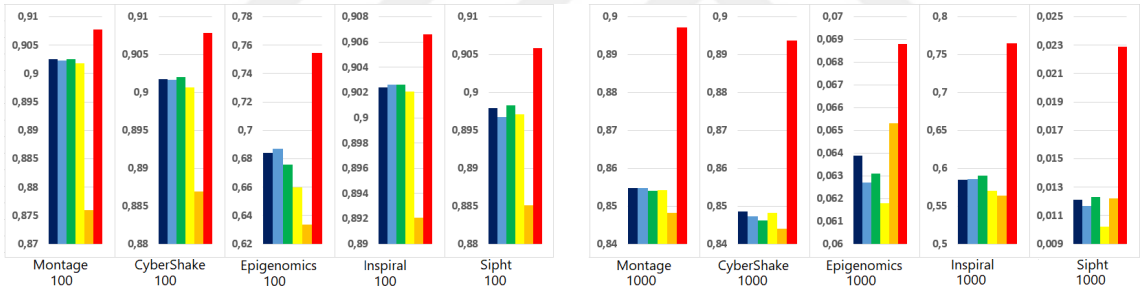
In this experiment, the response of the algorithms is validated when the number of objectives is updated, which is the second form of dynamism considered for the workflow scheduling problem in cloud computing. Three different objectives are initially considered, which are the minimization of makespan and cost and the maximization of reliability. The minimization of energy, the maximization of utilization and the minimization of degree of imbalance are the objectives that are added to or discarded from the set of the objectives as the fourth, fifth and sixth ones. The number of objectives $o(t)$, changes one at a time for each distinct workflow scheduling period t as follows;



(a) *NS*



(b) *SS*



(c) *HV*

Figure 6.2: Performance of algorithms in (a) *NS*, (b) *SS* and (c) *HV* metrics for average uncertainty

$$o(t) = o(t+6) = \begin{cases} 3 & t = 0 \\ o(t-1)+1 & t \in [1,3] \\ o(t-1)-1 & t \in [4,6] \end{cases} \tag{6.1}$$

The minimum number of optimization objectives is specified as three and the maximum number of optimization objectives as six for this category of the experiments for workflow scheduling problem in cloud computing. To simulate the changing number of objectives

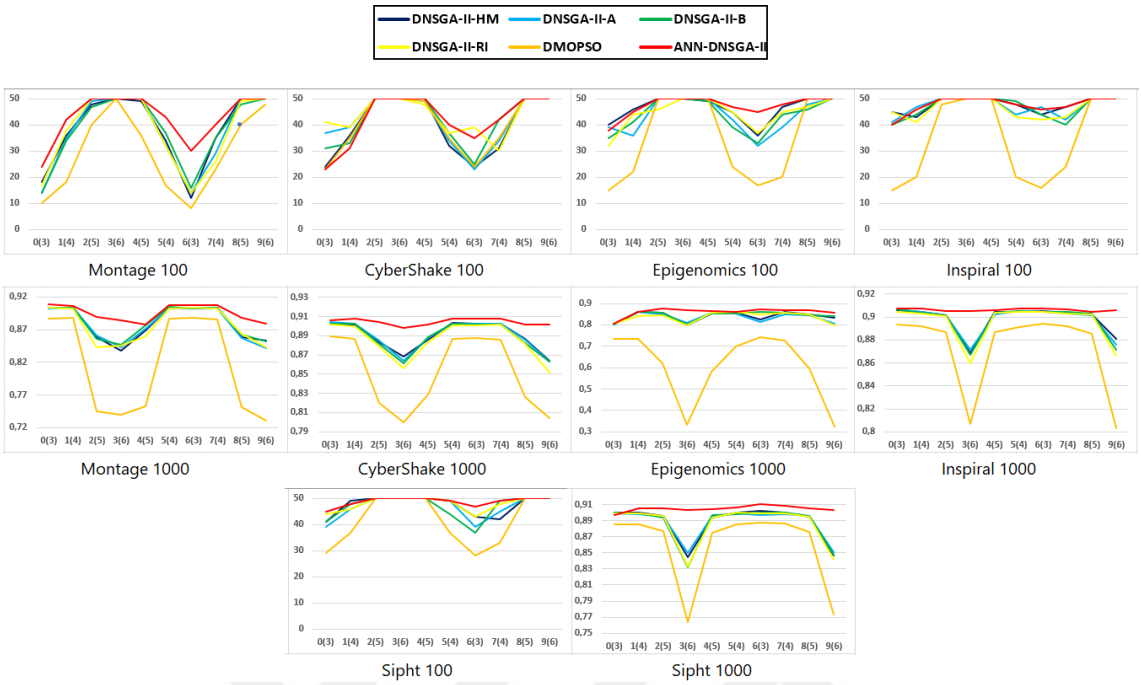
due to changing workflow execution requirements, the number of objectives changes from three to six and then from six to three one by one, which causes periodical fluctuation at every six discrete time t.

The results of the reactions of the algorithms to the expansion and the contraction of the objective space of the problem are presented in Figure 6.3. In the horizontal axis of the figure, the first number refers to the number of changes and the second number between parenthesis refers to the number of objectives used in that period. In the vertical axis of the figure, the performance values of the algorithms in the given metrics are shown. It is observed that the algorithms follow certain patterns while the objective space is exposed to the expansion and contraction. In Figure 6.3(a), the number of found non-dominated solutions increases in the expansion of the objective space, since the domination among solutions in the population is getting harder. In other words, the possibility of a solution to dominate other solutions in every objective is relatively low when the number of objectives available is high. The number of non-dominated solutions push the limit of the archive, which is 50 in our experiments, when number of objectives is 6 in most cases.

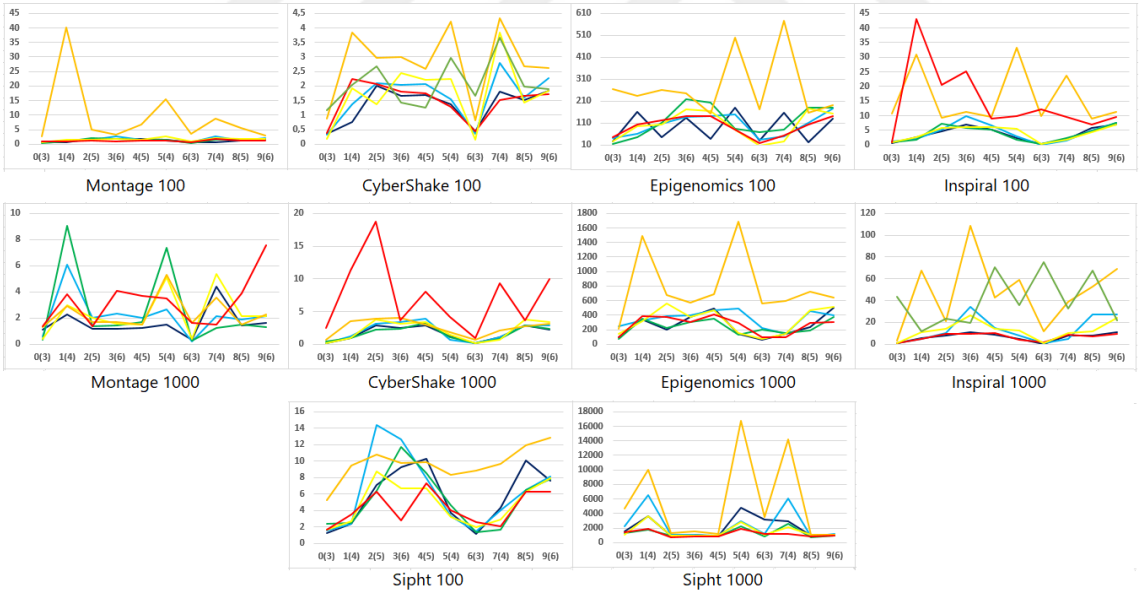
In Figure 6.3(b), the algorithms that are sensitive to *SS* metric such as NN-DNSGA-II and DMOPSO, fluctuate more than the other algorithms. From those results, it can be inferred that the algorithms that are especially worse at finding well-distributed solutions are exposed to more fluctuation in *SS*. The performance of the other algorithms are nearly equal to each other in the most cases in *SS* metric. In Figure 6.3(c), when the number of objectives decreases from 6 to 3 in the periods from 3 to 6, the performances of the algorithms gradually improves for converging close to POF.

On the other hand, when the number of objectives increases from 3 to 6 in the periods from 6 to 9, they become gradually worse, which is associated with the absence of sufficient diversity in the population due to expansion of POF. While the performances of the algorithms are at the best level in the periods 3 and 9, where the maximum number of objectives is reached, they are at the worst level in the period 6, where the minimum number of objectives is reached. Among the algorithms, the poorest results in this metric belong to DMOPSO, especially for workflows with 100 tasks. On the other hand, the performance of the NN-DNSGA-II algorithm in this metric stays almost unaffected. NN-DNSGA-II algorithm outperforms the other alternatives with respect to *NS* and *HV*,

while there is no apparent winner with respect to *SS*.



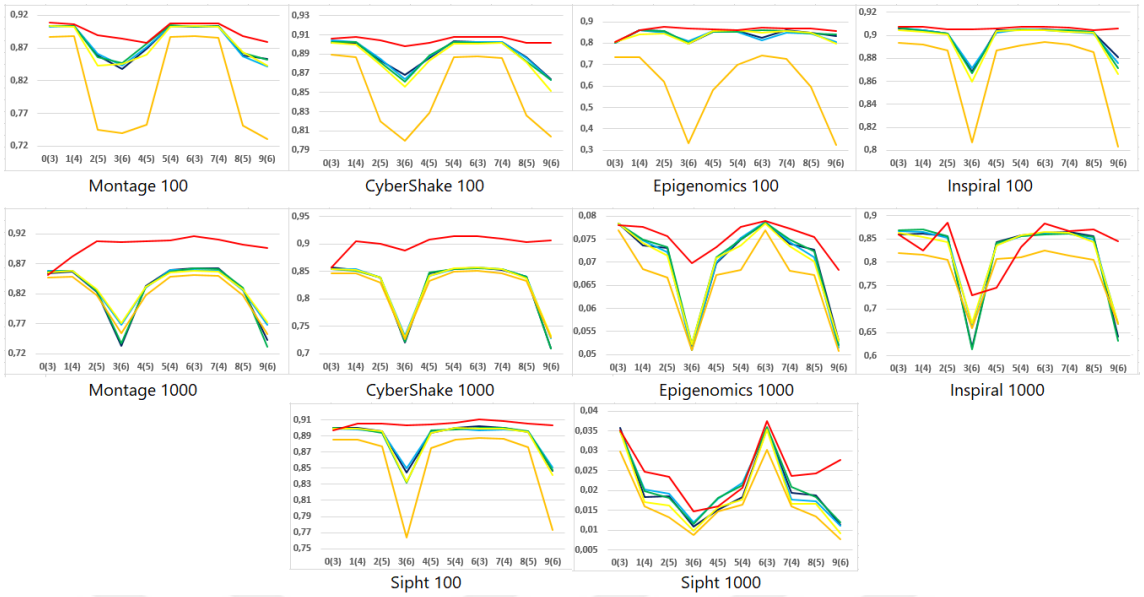
(a) *NS*



(b) *SS*

6.2.1. Further Analysis of Changing Number of Objectives

The frequency of changes and severity of changes are updated in this experiment while changing the number of objectives dynamically. In the first test, 100 and 500 generations per change for high and low frequency cases are considered, respectively.



(c) *HV*

Figure 6.3: Performance of algorithms in (a) *NS*, (b) *SS* and (c) *HV* metrics for changing number of objectives

The empirical results are provided in Table 6.5 and Table 6.6, where all algorithms show better performance in all metrics when the objectives change rather slowly, i.e., $f = 500$. As happened in the experiments for frequency of changes in resource failures, the algorithms are better when scheduling smaller and less complex workflows. For instance, the performance difference of the same algorithms between Epigenomics 100 and Epigenomics 1000 is clearly seen. The NN-DNSGA-II algorithm obtains the best *NS* and *HV* values under both frequency of changes for all workflows. On the other hand, it still suffers from finding well-distributed solutions for the workflows except from Montage 100, Epigenomics 100 and some cases for Sipht 100. Overall, the NN-DNSGA-II algorithm outperforms the other algorithms for 25 out of 30 cases and 22 out of 30 cases for workflows with 100 and 1000 tasks, respectively.

Our second and final test measures the effectiveness of the algorithms when severity of changes is varied, while updating the number of objectives considered. For low severity case, the number of objectives change according to the sequence presented in the Equation 6.2, where one objective increases or decreases at a time. For high severity case, three objectives increase and decrease at a time with the following sequence:

Table 6.5: Varying Frequency of Change in Changing Number of Objectives On Workflows With 100 Tasks

		Montage 100		CyberShake 100		Epigenomics 100		Inspiral 100		Sipht 100	
Algorithm	Metric	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$
DNSGA-II-HM	NS	38.6	44.6	39.9	41.9+	45.9	47.4+	47.3	48.6+	47.8+	49.2+
	SS	1.32	0.92	**1.30**	**1.32**	105.84	100.99	**3.55**	3.59	5.83	4.20
	HV	0.8794+	0.8948	0.8901+	0.8992	0.8389+	0.8474+	0.8982	0.9031+	0.8878+	0.8966+
DNSGA-II-A	NS	37.3+	44.7	41.7	41.7+	45.2	47.8+	47.5	47.9+	47.1+	49.2+
	SS	1.50	0.95	1.62	1.40	106.72	111.32	4.05	**3.49**	6.02	4.17
	HV	0.8792+	0.8944	0.0.8911+	0.9002	0.8403+	0.8599+	0.8998	0.9027+	0.8862+	0.8952+
DNSGA-II-B	NS	38.6	45.1	41.6	42.2+	44.2+	48.4	46.7	49.0	47.0+	49.5+
	SS	1.23	0.92	1.32	1.57	112.50	129.54	3.87	4.62	5.02	4.99
	HV	0.8816	0.8949	0.8890+	0.8992	0.8500	0.8569+	0.8965	0.9049+	0.8880+	0.8969+
DNSGA-II-RI	NS	37.6+	44.3	40.9	44.0	45.2	48.1	46.7	49.4	48.8	49.7+
	SS	1.60	1.06	1.76	1.50	110.91	108.42	5.12	3.53	4.90	**3.77**
	HV	0.8770+	0.8930	0.8887+	0.8985	0.8327+	0.8462+	0.8953	0.9000+	0.8842+	0.8962+
DMOPSO	NS	28.7+	33.0+	39.4+	43.5	33.0+	34.5+	39.2+	37.9+	41.6+	44.7+
	SS	10.56+	6.44+	3.96+	2.68+	300.65+	300.74+	15.24+	8.45+	11.62+	8.89+
	HV	0.8055+	0.8233+	0.8567+	0.8549+	0.6244+	0.7354+	0.8603+	0.8712+	0.8577+	0.8732+
NN-DNSGA-II	NS	**42.9**	**46.0**	**41.9**	**45.9**	**47.0**	**49.2**	**48.0**	**49.7**	**48.9**	**50.0**
	SS	**1.12**	**0.90**	2.07	2.00	**83.12**	**86.66**	15.13+	9.56+	**4.12**	4.55
	HV	**0.8956**	**0.8988**	**0.9041**	**0.9079**	**0.8612**	**0.8778**	**0.9034**	**0.9184**	**0.9054**	**0.9102**

Table 6.6: Varying Frequency of Change in Changing Number of Objectives On Workflows With 1000 Tasks

		Montage 1000		CyberShake 1000		Epigenomics 1000		Inspiral 1000		Sipht 1000	
Algorithm	Metric	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$	$f = 100$	$f = 500$
DNSGA-II-HM	NS	34.5	39.0	48.5	49.2	36.8	39.7	44.0	46.2	32.4	32.4
	SS	**1.63**	**1.51**	1.67	2.41	259.45	257.82	6.67	**5.30**	2081.81	2591.77
	HV	0.8257	0.8375	0.8231	0.8327	0.0699	0.0723	0.8124	0.8286	0.0202	0.0233
DNSGA-II-A	NS	34.3	39.4	47.4	49.5	35.5	40.0	42.8	48.0	31.6	32.6
	SS	2.21	1.87	1.93	2.69	353.39	270.10	13.40	11.68	2413.17	1506.57
	HV	0.8313	0.8426	0.8258	0.8387	0.0698	0.0723	0.8183	0.8336	0.0208	0.0234
DNSGA-II-B	NS	35.5	37.8	47.1	49.4	36.3	41.3	44.0	47.1	31.9	32.5
	SS	2.56	1.74	**1.65**	**2.19**	**234.72**	291.80	**6.51**	5.87	1316.50	1728.89
	HV	0.8253	0.8367	0.8230	0.8332	0.0703	0.0723	0.8114	0.8292	0.0211	0.0241
DNSGA-II-RI	NS	36.4	40.5	48.2	49.2	33.8	35.4	40.1	46.9	32.8	36.9
	SS	2.37	2.03	2.07	2.24	322.80	294.44	12.70	10.55	1572.91	**949.13**
	HV	0.8318	0.8359	0.8255	0.8330	0.0695	0.0722	0.8165	0.8300	0.0189	0.0213
DMOPSO	NS	34.9	37.3	43.2	45.7	30.2	31.3	33.3	35.4	30.1	30.8
	SS	2.34	2.05	2.55	2.30	787.53	669.62	47.69	40.29	5529.67	3630.77
	HV	0.8204	0.8213	0.8193	0.8198	0.0662	0.0668	0.7832	0.7870	0.0166	0.0178
NN-DNSGA-II	NS	**43.4**	**42.2**	**49.1**	**49.6**	**38.0**	**41.5**	**44.2**	**48.4**	**33.0**	**37.5**
	SS	3.24	3.19	7.22	6.10	263.39	**152.83**	40.26	26.27	**1165.63**	1098.74
	HV	**0.8996**	**0.8991**	**0.9007**	**0.9033**	**0.0749**	**0.0763**	**0.8340**	**0.8746**	**0.0246**	**0.0254**

$$o(t) = o(t+2) = \begin{cases} 3 & t = 0 \\ o(t-1)+3 & t \in [1] \\ o(t-1)-3 & t \in [2] \end{cases} \tag{6.2}$$

The experimental results for severity of changes while updating the number of objectives considered in the optimization are presented in Table 6.7 and Table 6.8. Those results show that the performances of all algorithms are better when the dimension of the objective space slowly changes since it is easier for them to adapt to slowly changing number of objectives. While DMOPSO is the worst algorithms in all types of metrics, NN-DNSGA-II outperforms the other algorithms for 22 out of 30 cases and 21 out of 30 cases for workflows with 100 and 1000, respectively. It mainly dominates based on *NS* and *HV* metrics as happened in other experiments. Overall, it is observed in all experiments that NN-DNSGA-II out- performs the other five alternatives for most of the cases in three metrics. According to the statistical tests, it is significantly better than others at least one metric in each case.

Table 6.7: Varying Severity of Change in Changing Number of Objectives On Workflows With 100 Tasks

		Montage 100		CyberShake 100		Epigenomics 100		Inspiral 100		Sipht 100	
Algorithm	Metric	$s=1$	$s=3$	$s=1$	$s=3$	$s=1$	$s=3$	$s=1$	$s=3$	$s=1$	$s=3$
DNSGA-II-HM	NS	38.6	33.0+	39.9	39.5	45.9	41.8+	47.3	45.0	47.8	43.9+
	SS	1.32	**1.12**	**1.30**	1.10	105.84	96.78	**3.55**	4.02	5.83	**5.03**
	HV	0.8794+	0.8722+	0.8901+	0.8765+	0.8389+	0.7952+	0.8982	0.8899+	0.8878+	0.8733+
DNSGA-II-A	NS	37.3+	34.1+	41.7	39.0	45.2	41.6+	47.5	**46.0**	47.1+	44.3+
	SS	1.50	1.34	1.62	**1.07**	106.72	**69.88**	4.05	**3.22**	6.02	5.59
	HV	0.8792+	0.8715+	0.0.8911+	0.8800+	0.8403+	0.7912+	0.8998	0.8900+	0.8862+	0.8686+
DNSGA-II-B	NS	38.6	31.9+	41.6	40.2	44.2+	42.5	46.7	44.1	47.0+	45.3
	SS	1.23	1.29	1.32	1.15	112.50	99.67	3.87	4.02	5.02	5.99
	HV	0.8816	0.8733+	0.8890+	0.8852+	0.8500	0.8092+	0.8965	0.8932	0.8880+	0.8784+
DNSGA-II-RI	NS	37.6+	33.2+	40.9	41.5	45.2	43.7	46.7	44.1	48.8	45.2
	SS	1.60	1.57	1.76	1.21	110.91	111.32+	5.12	3.99	4.90	6.97
	HV	0.8770+	0.8677+	0.8887+	0.8773+	0.8327+	0.7699+	0.8953	0.8850+	0.8842+	0.8632+
DMOPSO	NS	28.7+	29.7+	39.4+	36.8+	33.0+	33.0+	39.2+	33.7+	41.6+	37.5+
	SS	10.56+	2.97+	3.96+	1.93	300.65+	267.43+	15.24+	8.56+	11.62+	12.28+
	HV	0.8055+	0.7943+	0.8567+	0.8367+	0.6244+	0.5713+	0.8603+	0.8500+	0.8577+	0.8412+
NN-DNSGA-II	NS	**42.9**	**39.7**	**41.9**	**42.6**	**47.0**	**44.5**	**48.0**	43.2	**48.9**	**47.7**
	SS	**1.12**	1.20	2.07	3.11+	**83.12**	100.63	15.13+	11.99+	**4.12**	12.09+
	HV	**0.8956**	**0.9016**	**0.9041**	**0.9095**	**0.8612**	**0.8565**	**0.9034**	**0.9056**	**0.9054**	**0.9022**

Table 6.8: Varying Severity of Change in Changing Number of Objectives On Workflows With 1000 Tasks

		Montage 1000		CyberShake 1000		Epigenomics 1000		Inspiral 1000		Sipht 1000	
Algorithm	Metric	$s=1$	$s=3$	$s=1$	$s=3$	$s=1$	$s=3$	$s=1$	$s=3$	$s=1$	$s=3$
DNSGA-II-HM	NS	34.5	31.7	48.5	45.0	36.8	28.6	44.0	41.4	32.4	28.7
	SS	**1.63**	0.90	1.67	**1.20**	259.45	248.68	6.67	5.10	2081.81	1203.44
	HV	0.8257	0.7969	0.8231	0.7842	0.0699	0.0651	0.8124	0.7454	0.0202	0.0235
DNSGA-II-A	NS	34.3	31.8	47.4	44.5	35.5	28.9	42.8	39.6	31.6	28.2
	SS	2.21	1.12	1.93	1.56	353.39	306.15	13.40	14.22	2413.17	**1121.76**
	HV	0.8313	0.8116	0.8258	0.7945	0.0698	0.0655	0.8183	0.7699	0.0208	0.0216
DNSGA-II-B	NS	35.5	33.2	47.1	45.2	36.3	28.9	44.0	43.5	31.9	28.6
	SS	2.56	**0.81**	**1.65**	1.29	**234.72**	**206.00**	**6.51**	**4.87**	1316.50	1457.71
	HV	0.8253	0.7972	0.8230	0.7836	0.0703	0.0651	0.8114	0.7501	0.0211	0.0229
DNSGA-II-RI	NS	36.4	33.2	48.2	44.9	33.8	29.3	40.1	41.5	32.8	29.0
	SS	2.37	1.10	2.07	1.73	322.80	339.14	12.70	15.36	1572.91	1279.77
	HV	0.8318	0.8119	0.8255	0.7936	0.0695	0.0651	0.8165	0.7719	0.0189	0.0214
DMOPSO	NS	34.9	32.9	43.2	38.6	30.2	27.7	33.3	32.8	30.1	28.7
	SS	2.34	1.40	2.55	1.92	787.53	527.68	47.69	42.69	5529.67	2569.63
	HV	0.8204	0.8031	0.8193	0.7892	0.0662	0.0640	0.7832	0.7400	0.0166	0.0190
NN-DNSGA-II	NS	**43.4**	**40.1**	**49.1**	**45.3**	**38.0**	**31.1**	**44.2**	**43.8**	**33.0**	**29.9**
	SS	3.24	3.92	7.22	10.63	263.39	248.64	40.26	33.36	**1165.63**	1378.21
	HV	**0.8996**	**0.8936**	**0.9007**	**0.8936**	**0.0749**	**0.0719**	**0.8340**	**0.8526**	**0.0246**	**0.0299**

7. CONCLUSIONS AND FUTURE WORK

Workflow scheduling problem is one among the mostly-studied problems for cloud computing in the literature. It is more realistic and challenging when it is addressed in dynamic environments. To the best our knowledge, our work is one of the first systematic attempts in the literature in order to model the dynamic workflow scheduling problem as a dynamic multi-objective optimization problem (DMOP). The sources of dynamism in the problem are primarily driven by changing set of resources due to failures and changing number of objectives due to a set of real-world scenarios encountered during workflow executions. The first type of dynamism changes the structure of the decision space of the problem, where the tasks must be rescheduled to available resources. On the other hand, the second type of dynamism changes the structure of the objective space of the problem by expansion and contraction, where the diversity and convergence issues must be handled.

The dynamic workflow scheduling problem in cloud computing as a dynamic multi-objective optimization problem incorporates into different optimization objectives. The minimization of makespan, cost, energy and degree of imbalance, and the maximization of reliability and utilization are the those optimization goals considered in this thesis. For the given problem, five state-of-art non-prediction based multi-objective evolutionary algorithms (the DNSGA-II-A, the DNSGA-II-B, the DNSGA-II-HM, the DNSGA-II-RI and the DMOPSO algorithms) are adapted. Secondly, a novel neural network based dynamic multi-objective evolutionary algorithm (NN-DNSGA-II) is proposed, which targets to exploit history of optimal solutions for estimating future optimal solutions after changes. The performance evaluation of algorithms are based on ten different workflow instances from the Pegasus Workflow Management System and technical specifications of the resources from Amazon EC2. The experimental results validates the applicability of our NN-DNSGA-II algorithm for dynamic workflow scheduling problem. Specifically, it significantly outperforms the other alternatives for most of the test instances with respect to the three metrics that are used for DMOPs, the number of non-dominated solutions, the Schott's spacing and the Hypervolume metric.

As a future work, it is planned to extend this study by integrating with other machine learning techniques in addition to neural networks including support vector

machines and recurrent neural networks. Another future direction is to add novel dynamic multi-objective optimization algorithms based on multiple population and memory-based techniques to the existing framework. In addition, the algorithms that specifically target to resolve the challenges of changing multi-dimensional objective space due to the changing number of objectives can be proposed for the problem. Finally, a more realistic cloud computing model that supports multiple virtual machines in resources and multiple data centres provided by multiple cloud providers can be studied in the future.

REFERENCES

[1] Zhaomeng Zhu et al. “Evolutionary Multi-Objective Workflow Scheduling in Cloud”. In: *IEEE Trans. Parallel Distrib. Syst.* 27.5 (2016), pp. 1344–1357.

[2] Dalibor Klusácek et al. “Scheduling Scientific Workloads in Private Cloud: Problems and Approaches”. In: *Proceedings of the 10th International Conference on Utility and Cloud Computing, UCC 2017, Austin, TX, USA, December 5-8, 2017*. 2017, pp. 9–18.

[3] Ehab Nabiel Alkhanak and Sai Peck Lee. “A hyper-heuristic cost optimisation approach for Scientific Workflow Scheduling in cloud computing”. In: *Future Generation Comp. Syst.* 86 (2018), pp. 480–506.

[4] Duncan A. Brown et al. “A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis”. In: *Workflows for e-Science: Scientific Workflows for Grids*. London: Springer London, 2007, pp. 39–59.

[5] Gabrielle Allen et al. “Towards an integrated GIS-based coastal forecast workflow”. In: *Concurrency and Computation: Practice and Experience* 20.14 (2008), pp. 1637–1651.

[6] Renzhi Chen, Ke Li, and Xin Yao. “Dynamic Multiobjectives Optimization With a Changing Number of Objectives”. In: *IEEE Trans. Evolutionary Computation* 22.1 (2018), pp. 157–171.

[7] Alexandru Iosup et al. “On the dynamic resource availability in grids”. In: *8th IEEE/ACM International Conference on Grid Computing (GRID 2007), September 19-21, 2007, Austin, Texas, USA, Proceedings*. 2007, pp. 26–33.

[8] Ming Mao and Marty Humphrey. “A Performance Study on the VM Startup Time in the Cloud”. In: *2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, June 24-29, 2012*. 2012, pp. 423–430.

[9] *Amazon Web Service*. `https://aws.amazon.com/ec2/`.

[10] Jurgen Branke. *Evolutionary Optimization in Dynamic Environments*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[11] Shengxiang Yang and Xin Yao. *Evolutionary Computation for Dynamic Optimization Problems*. Springer Publishing Company, 2013.

[12] M. Farina, K. Deb, and P. Amato. "Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications". In: *Trans. Evol. Comp* 8.5 (2004), pp. 425–442.

[13] Yaochu Jin and J. Branke. "Evolutionary Optimization in Uncertain Environments-a Survey". In: *Trans. Evol. Comp* 9.3 (2005), pp. 303–317.

[14] Frank Vavak, Ken Jukes, and Terence C. Fogarty. "Adaptive Combustion Balancing in Multiple Burner Boiler Using a Genetic Algorithm with Variable Range of Local Search". In: *Proceedings of the 7th International Conference on Genetic Algorithms, East Lansing, MI, USA, July 19-23, 1997*. 1997, pp. 719–726.

[15] John J. Grefenstette. "Genetic Algorithms for Changing Environments". In: *Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28-30, 1992*. 1992, pp. 139–146.

[16] Changhe Li, Shengxiang Yang, and Ming Yang. "An Adaptive Multi-swarm Optimizer for Dynamic Optimization Problems". In: *Evol. Comput.* 22.4 (2014), pp. 559–594.

[17] Juergen Branke. "Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems". In: vol. 3. Feb. 1999, 1882 Vol. 3.

[18] Arrchana Muruganantham, Kay Chen Tan, and Prahlad Vadakkepat. "Evolutionary Dynamic Multiobjective Optimization Via Kalman Filter Prediction". In: *IEEE Trans. Cybernetics* 46.12 (2016), pp. 2862–2873.

[19] Almuth Meier and Oliver Kramer. "Recurrent neural network-predictions for PSO in dynamic optimization". In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2018, Kyoto, Japan, July 15-19, 2018*. 2018, pp. 29–36.

[20] Maria Alejandra Rodriguez and Rajkumar Buyya. "Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds". In: *IEEE Trans. Cloud Computing* 2.2 (2014), pp. 222–235.

[21] Li Liu et al. “Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing”. In: *Concurrency and Computation: Practice and Experience* 29.5 (2017).

[22] Hamid Mohammadi Fard et al. “A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments”. In: *12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid 2012, Ottawa, Canada, May 13-16, 2012*. 2012, pp. 300–309.

[23] Miao Zhang et al. “An adaptive multi-objective evolutionary algorithm for constrained workflow scheduling in Clouds”. In: *Distributed and Parallel Databases* 36.2 (2018), pp. 339–368.

[24] Zong-Gan Chen et al. “Multiobjective Cloud Workflow Scheduling: A Multiple Populations Ant Colony System Approach”. In: *IEEE Transactions on Cybernetics* PP (May 2018), pp. 1–15.

[25] Xiumin Zhou et al. “Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT”. In: *Future Generation Computer Systems* 93 (2019), pp. 278–289.

[26] Suraj Pandey et al. “A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments”. In: *24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13 April 2010*. 2010, pp. 400–407.

[27] Wei-neng Chen and Jun Zhang. “A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints”. In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, SMC 2012, Seoul, Korea (South), October 14-17, 2012*. 2012, pp. 773–778.

[28] Shubham et al. “An effective multi-objective workflow scheduling in cloud computing: A PSO based approach”. In: *2016 Ninth International Conference on Contemporary Computing (IC3)* (2016), pp. 1–6.

[29] Juan José Durillo, Vlad Nae, and Radu Prodan. “Multi-objective Workflow Scheduling: An Analysis of the Energy Efficiency and Makespan Tradeoff”. In:

13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013, Delft, Netherlands, May 13-16, 2013. 2013, pp. 203–210.

[30] Xiaofeng Wang et al. “Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm”. In: *Future Generation Comp. Syst.* 27.8 (2011), pp. 1124–1134.

[31] Zhenyu Wen et al. “Cost Effective, Reliable and Secure Workflow Deployment over Federated Clouds”. In: *IEEE Trans. Services Computing* 10.6 (2017), pp. 929–941.

[32] Sonia Yassa et al. “Multi-Objective Approach for Energy-Aware Workflow Scheduling in Cloud Computing Environments”. In: *TheScientificWorldJournal*. 2013.

[33] Bhaskar Prasad Rimal and Martin Maier. “Workflow Scheduling in Multi-Tenant Cloud Computing Environments”. In: *IEEE Trans. Parallel Distrib. Syst.* 28.1 (2017), pp. 290–304. DOI: 10.1109/TPDS.2016.2556668. URL: https://doi.org/10.1109/TPDS.2016.2556668.

[34] Huangke Chen et al. “Uncertainty-Aware Real-Time Workflow Scheduling in the Cloud”. In: *9th IEEE International Conference on Cloud Computing, CLOUD 2016, San Francisco, CA, USA, June 27 - July 2, 2016*. 2016, pp. 577–584.

[35] Huangke Chen et al. “Real-time workflows oriented online scheduling in uncertain cloud environment”. In: *The Journal of Supercomputing* 73.11 (2017), pp. 4906–4922.

[36] Hamid Mohammadi Fard, Radu Prodan, and Thomas Fahringer. “A Truthful Dynamic Workflow Scheduling Mechanism for Commercial Multicloud Environments”. In: *IEEE Trans. Parallel Distrib. Syst.* 24.6 (2013), pp. 1203–1212.

[37] Mohammad Masdari et al. “Towards workflow scheduling in cloud computing: A comprehensive analysis”. In: *J. Network and Computer Applications* 66 (2016), pp. 64–82.

[38] Maria Alejandra Rodriguez and Rajkumar Buyya. “A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing

environments". In: *Concurrency and Computation: Practice and Experience* 29.8 (2017).

[39] Kassian Plankensteiner, Radu Prodan, and Thomas Fahringer. "A New Fault Tolerance Heuristic for Scientific Workflows in Highly Distributed Environments Based on Resubmission Impact". In: *Fifth International Conference on e-Science, e-Science 2009, 9-11 December 2009, Oxford, UK*. 2009, pp. 313–320.

[40] Yang Zhang et al. "Combined Fault Tolerance and Scheduling Techniques for Workflow Applications on Computational Grids". In: *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2009, Shanghai, China, 18-21 May 2009*. 2009, pp. 244–251.

[41] Kassian Plankensteiner and Radu Prodan. "Meeting Soft Deadlines in Scientific Workflows Using Resubmission Impact". In: *IEEE Trans. Parallel Distrib. Syst.* 23.5 (2012), pp. 890–901.

[42] Jialei Liu et al. "Using Proactive Fault-Tolerance Approach to Enhance Cloud Service Reliability". In: *IEEE Trans. Cloud Computing* 6.4 (2018), pp. 1191–1202.

[43] Sheng Uei Guan, Qian Chen, and Wenting Mo. "Evolving Dynamic Multi-Objective Optimization Problems with Objective Replacement". In: *Artif. Intell. Rev.* 23.3 (2005), pp. 267–293.

[44] Liang Huang, Il Hong Suh, and Ajith Abraham. "Dynamic multi-objective optimization based on membrane computing for control of time-varying unstable plants". In: *Inf. Sci.* 181.11 (2011), pp. 2370–2391.

[45] Radhia Azzouz, Slim Bechikh, and Lamjed Ben Said. "A Multiple Reference Point-based evolutionary algorithm for dynamic multi-objective optimization with undetectable changes". In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*. 2014, pp. 3168–3175.

[46] Zong-Gan Chen et al. "Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm". In: *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*. 2015, pp. 708–714.

[47] Heiner Zille, Andre Kottenhahn, and Sanaz Mostaghim. “Dynamic Distance Minimization Problems for dynamic multi-objective optimization”. In: *2017 IEEE Congress on Evolutionary Computation, CEC 2017, Donostia, San Sebastián, Spain, June 5-8, 2017*. 2017, pp. 952–959.

[48] Maria Carla Calzarossa, Marco L. Della Vedova, and Daniele Tessera. “A methodological framework for cloud resource provisioning and scheduling of data parallel applications under uncertainty”. In: *Future Generation Computer Systems* 93 (2019), pp. 212–223.

[49] Kalyanmoy Deb, Udaya Bhaskara Rao N., and S. Karthik. “Dynamic Multi-objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-thermal Power Scheduling”. In: *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, Matsushima, Japan, March 5-8, 2007, Proceedings*. 2006, pp. 803–817.

[50] Min Liu et al. “An adaptive diversity introduction method for dynamic evolutionary multiobjective optimization”. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*. 2014, pp. 3160–3167.

[51] Chi Keong Goh and Kay Chen Tan. “A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization”. In: *IEEE Trans. Evolutionary Computation* 13.1 (2009), pp. 103–127.

[52] Mardé Greeff and Andries Petrus Engelbrecht. “Solving dynamic multi-objective problems with vector evaluated particle swarm optimisation”. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC2008, June 1-6, 2008, Hong Kong, China*. 2008, pp. 2917–2924.

[53] Yu Wang and Bin Li. “Investigation of memory-based multi-objective optimization evolutionary algorithm in dynamic environment”. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009, Trondheim, Norway, 18-21 May, 2009*. 2009, pp. 630–637.

[54] Zhou Peng, Jinhua Zheng, and Juan Zou. “A population diversity maintaining strategy based on dynamic environment evolutionary model for dynamic multiobjective optimization”. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*. 2014, pp. 274–281.

[55] Shaaban Sahmoud and Haluk Rahmi Topcuoglu. “A Memory-Based NSGA-II Algorithm for Dynamic Multi-objective Optimization Problems”. In: *Applications of Evolutionary Computation - 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings, Part II*. 2016, pp. 296–310.

[56] Ruochen Liu et al. “A Novel Cooperative Coevolutionary Dynamic Multi-objective Optimization Algorithm Using a New Predictive Model”. In: *Soft Comput.* 18.10 (2014), pp. 1913–1929.

[57] Ruochen Liu, Jing Fan, and Licheng Jiao. “Integration of Improved Predictive Model and Adaptive Differential Evolution Based Dynamic Multi-objective Evolutionary Optimization Algorithm”. In: *Applied Intelligence* 43.1 (2015), pp. 192–207.

[58] Xiao Fang Liu, Zhi-Hui Zhan, and Jun Zhang. “Neural Network for Change Direction Prediction in Dynamic Optimization”. In: *IEEE Access* 6 (2018), pp. 72649–72662.

[59] Min Jiang et al. “Solving dynamic multi-objective optimization problems via support vector machine”. In: *Tenth International Conference on Advanced Computational Intelligence, ICACI 2018, Xiamen, China, March 29-31, 2018*. 2018, pp. 819–824.

[60] Carlos A. Coello Coello, Gregorio Toscano Pulido, and M. Salazar Lechuga. “Handling Multiple Objectives With Particle Swarm Optimization”. In: *IEEE Trans. Evolutionary Computation* 8.3 (2004), pp. 256–279.

[61] Kalyanmoy Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Trans. Evolutionary Computation* 6.2 (2002), pp. 182–197.

[62] Betül Demiröz and Haluk Rahmi Topcuoglu. “Static Task Scheduling with a Unified Objective on Time and Resource Domains”. In: *Comput. J.* 49.6 (2006), pp. 731–743.

[63] Zhangjun Wu et al. “A market-oriented hierarchical scheduling strategy in cloud workflow systems”. In: *The Journal of Supercomputing* 63.1 (2013), pp. 256–293.

[64] Sathiya Bharathi et al. “Characterization of scientific workflows”. In: *2008 Third Workshop on Workflows in Support of Large-Scale Science* (2008), pp. 1–10.

[65] Shouyong Jiang and Shengxiang Yang. “A framework of scalable dynamic test problems for dynamic multi-objective optimization”. In: *2014 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, CIDUE 2014, Orlando, FL, USA, December 9-12, 2014*. 2014, pp. 32–39.

[66] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. “Comparison of Multiobjective Evolutionary Algorithms: Empirical Results”. In: *Evolutionary Computation* 8.2 (2000), pp. 173–195.

[67] Hisao Ishibuchi et al. “Many-objective Test Problems to Visually Examine the Behavior of Multiobjective Evolution in a Decision Space”. In: *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II*. Springer-Verlag, 2010, pp. 91–100.

[68] F. Wilcoxon. “Individual comparisons by ranking methods”. In: *Biometrics Bull.* 1.6 (1945), pp. 80–83.

RESUME

GOSHGAR ISMAYILOV

Marmara University

Computer Engineering Department, Faculty of Engineering,

Goztepe Campus, Kadikoy, Istanbul, Turkey

Phone (Cell) +90-5395039974

e-mail: goshgarismayilov@marun.edu.tr

EDUCATION

M. S., Computer Science Marmara University, Faculty of Engineering, Istanbul, Turkey, 2019.
Thesis Topic: Dynamic Multi-Objective Workflow Scheduling in Cloud Computing, *Advisor:* Prof. Haluk Rahmi TOPCUOGLU.

B. S., Computer Science Marmara University, Istanbul, Turkey, 2016.
Thesis Topic: Evolutionary Techniques for Dynamic Traveling Salesman Problem, *Advisor:* Prof. Haluk Rahmi TOPCUOGLU.

WORK EXPERIENCE

April 2019 – present:

Software Developer, BAYKAR Savunma, Istanbul, Turkey

December 2016 – present:

Researcher, Computer Engineering, Marmara University, Istanbul, Turkey

June 2015 – September 2015:

Intern Researcher, Computer Engineering, Marmara University, Istanbul, Turkey

June 2014 – September 2014:

Intern Software Developer, Borda Technology, Istanbul, Turkey

PUBLICATIONS

1. G. Ismayilov, H. R. Topcuoglu, Dynamic multi-objective workflow scheduling for cloud computing based on evolutionary algorithms, in: 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion, UCC Companion 2018, Zurich, Switzerland, December 17-20, 2018, pp. 103-108.
2. G. Ismayilov, H. R. Topcuoglu, Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing, Future Generation Computer Systems, (accepted).

RESEARCH INTERESTS

Dynamic Optimization Problems, Evolutionary Algorithms, Cloud Computing, Unmanned Aerial Vehicles

FOREIGN LANGUAGES

Azerbaijani (Native), Turkish (Native), English (Fluent)