MULTI-OBJECTIVE TASK SCHEDULING IN HETEROGENEOUS FOG ENVIRONMENTS

by

Lokman Altın

B.S., Computer Engineering, Marmara University, 2012

M.S., Computer Engineering, Marmara University, 2015

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis advisors Prof. Fikret Gürgen and Prof. Haluk Topçuoğlu for their support and encouragement throughout this study. I would also thank to Prof. Can Özturan, Prof. Deniz Turgay Altılar, Prof. Tunga Güngör and Assoc. Prof. Ali Haydar Özer for their participation in my thesis jury, their useful comments, and feedback.

I would like to thank my beloved wife, Damla Altın, for her support and understanding throughout this study.

ABSTRACT

MULTI-OBJECTIVE TASK SCHEDULING IN HETEROGENEOUS FOG ENVIRONMENTS

Limitations of the conventional cloud computing have surfaced as the internet evolves towards Future Internet where billions of devices connected to the global network producing enormous volume of data. Fog computing is proposed to overcome the drawbacks of the cloud computing which brings the computation and storage towards the edge of the network. Task scheduling on fog environments surges new challenges compared to scheduling on conventional cloud computing. Although there are a few recent work on task scheduling in fog computing, they are very limited and they do not represent most of the major challenges in fog computing. Various levels of heterogeneity and dynamism cause task scheduling problem to be more challenging for fog computing. In this thesis, we present a multi-objective task scheduling model with total of five objectives; and we propose two multi-objective multi rank scheduling algorithms for fog computing, the MOMRank and the LAMOMRank algorithms. The performance of the proposed strategies is assessed with well-known multi objective metaheuristics (the NSGA-II and the SPEA2 algorithms) and a widely used algorithm from the literature (the MOHEFT algorithm) using three common multi-objective metrics. Furthermore, set of highlighted individual metrics are also measured to address open issues in fog environments. We populated our workloads with the Pegasus workflows with dependent tasks that will produce network traffic and the DeFog applications that will demand real-time requirements. Additionally, we incorporate two task clustering schemes to the algorithms in order to improve data transmissions on interconnection networks. Results of empirical evaluations given in performance profiles over all instances validate significance of our algorithms in terms of multiobjective metrics, diminishing fog cluster network and reducing latency for real time applications.

ÖZET

HETEROJEN SİS ORTAMLARINDA ÇOK AMAÇLI GÖREV ZAMANLAMASI

İnternet, küresel ağa bağlı milyarlarca cihazın muazzam miktarda veri ürettiği *Geleceğin İnterneti*'ne doğru geliştikçe, geleneksel bulut bilişimin sınırlamaları ortaya çıktı. Sis bilişim, hesaplamayı ve depolamayı ağın ucuna doğru getiren bulut bilişimin dezavantajlarının üstesinden gelmek için önerilmiştir. Sis ortamlarında görev zamanlaması, geleneksel bulut bilişimde zamanlamaya kıyasla yeni zorluklar doğurur. Sis bilişimde görev zamanlaması üzerine yakın zamanda yapılmış birkaç çalışma olmasına rağmen, bunlar çok sınırlıdır ve sis bilişimdeki başlıca zorlukların çoğunu temsil etmezler. Çeşitli düzeylerde heterojenlik ve dinamizm, görev çizelgeleme probleminin sis hesaplama için daha zorlayıcı olmasına neden olur. Bu çalışmada, toplam beş hedefi olan çok amaçlı bir görev çizelgeleme modeli sunacağız ve sis hesaplama için çok amaçlı çok aşamalı iki çizelgeleme algoritması olan MOMRank ve LAMOMRank algoritmalarını öneriyoruz. Ortaya sürülen stratejinin performansı, iyi bilinen çok amaçlı meta-sezgisel (NSGA-II ve SPEA2 algoritmaları) ve literatürden yaygın olarak kullanılan bir algoritma (MOHEFT algoritması) ile üç yaygın çok amaçlı metrik kullanılarak değerlendirildi. Bunun yanında, öne çıkarılan bireysel metrikler ile de fog ortamlarındaki ağ tıkanıklığı ve görev bazlı gecikmenin ölçülmesi hedeflendi. Deney kümemizi birbirine bağımlı iş yükleri içeren Pegasus işakış yükleri ve gerçek zamanlı kısıtları olan DeFog uygulamaları ile oluşturduk. Ek olarak, ara bağlantı ağlarında veri iletimini iyileştirmek için iki görev kümeleme mekanizmasını algoritmalara dahil ediyoruz. Tüm problem örneklemlerinde verilen ampirik değerlendirmelerin sonuçları, veri aktarım maliyetlerini ve görev bazlı gecikmeyi azaltmak için hem algoritmamızın hem de entegre uzantıların önemini doğrulamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

C	Total cost
d_{mn}	Data movement requirement from task m to task n
D	Number of data transmission on workflow
$FT(t_m)$	Finish time of task m
L	Number of physical link on a fog cluster
N	Number of nodes
R	System reliability
T	Number of tasks
w_m	Workload of task m
y_{mj}	Allocation variable of task m to node j
$\Delta*$	Multi-objective spread metric
ϵ_j	Energy consumption of node j per unit time
ρ_j	Processing power of node j
τ_{ij}	Data transmission rate between nodes i and j
ζ	Total energy consumption

LIST OF ACRONYMS/ABBREVIATIONS

ARTT	Average Round Trip Time
CDC	Cloud Data Center
DAG	Directed Acyclic Graph
HV	Hypervolume
IoT	Internet of Things
IoE	Internet of Everything
LAMOMRank	Latency Aware Multi-objective Multi-Rank
MEC	Mobile Edge Computing
MOHEFT	Multi-objective Heterogeneous Earliest Finish Time
MOMRank	Multi-objective Multi-Rank
MOP	Multi-objective Optimization Problem
NSGA-II	Non-dominated Sorting Genetic Algorithm II
POF	Pareto Optimal Front
SPEA2	Strength Pareto Evolutionary Algorithm
WARSP	Weighted Average Response Time

1. INTRODUCTION

Limitations of conventional cloud computing have surfaced as the internet evolves in the era of the Internet of Everything (IoE) and 5G, where up to 1.2 trillion IoE devices are expected to be connected by 2030 producing an enormous volume of data [1]. Migrating this enormous volume of different-nature data to the cloud introduces challenges such as latency in the global network and security/privacy issues. Cloud computing also has limitations while responding low latency constrained real-time applications. These challenges can not be fully addressed with conventional centralized cloud computing paradigm. New cloud computing architecture is required to cope with such challenges. New distributed computing models such as Mobile Clouds [2], Mobile Edge Computing [3], Vehicular Networks [4], and Fog Computing [5] are surfaced in recent years. Fog computing is a novel decentralized computing approach introduced by Cisco to overcome the drawbacks of the cloud computing which brings the computation and storage towards the edge of the network [5]. Network devices such as routers, gateways are turned into computation and storage devices in the fog architecture. Beyond extension to the existing network components, local computation and storage cloudlets are also placed at the near edge of the network. The emerging fog computing paradigm has novel challenges such as uninterrupted service for mobile users and devices, security/privacy concerning multiple copies of user data, agile provisioning for quick response time, reliability, cost and energy [6].

Fog computing is a cloud paradigm that provides computing, storage and networking services between end users and the cloud. Edge and Fog usually used interchangeably. However fog computing paradigm differs from edge computing where processing nodes are not exclusively placed at edge of the network. Fog nodes are not only used for processing and storage as in edge nodes, but also used for managing, orchestrating, distributing and security [7]. Networking devices such as mobile base stations, gateways and routers are enabled for processing and storage services leading to multi-layer hierarchy [7]. The processing power and storage capacity increases

towards cloud in this hierarchy where more raw data processed towards edge of the network. Similar to the conventional Cloud architecture, Fog computing would also offer IaaS, PaaS and SaaS resources but not as a full Cloud Data Center (CDC). As the major advantage of Fog is the proximity to the end users to reduce latency, its expected there will be a few fog data centers per city. However as the business evolves, possible locations for fog data center ranges from local coffe shop to mobile cell towers as in the Mobile Edge Computing.

Fog nodes can communicate with different fog nodes via wired or wireless communication. In the multi-tier architecture of fog computing, fog nodes at the edge of the network might need less processing power, communication and storage compared to the higher level fog nodes. Main focus in the edge fog nodes is usually sensor activation, sensor control, sensor data collection and normalization. At the next higher level of fog hierarchy, raw data is filtered, transformed and processed. At near Cloud DC fog nodes aggregating data, forming information and finally knowledge. In case of large volume of data (Big Data) and enhanced analytics, edge fog nodes or the higher-tier fog nodes can be empowered with GPUs, FPGAs and DSPs to increase computational throughput. Storage devices of fog nodes also need to enable reliability and data integrity of varying fog computing tiers and nodes. Fog nodes can orchestrate applications in distributed fashion to provide load distribution, reliability, data sharing and reducing cloud communication [8].

Scaling of applications for different tiers of fog architecture is another advantage that enables only transmitting essential parts of the data by pre-processing. This would reduce the network traffic by reducing the data transmitted to CDCs. Workloads can be decomposed on CDCs and offloaded to the edge nodes, migrated by user devices to edge devices, or aggregated from sensors or devices on an edge node.

In emergence of Fog Computing, conventional centralized cloud architecture still preserves its importance. Cloud and edge nodes complete each other by enabling improvements for various types of applications. Some applications can be more favorable

to be executed in conventional central cloud while others well-fitted for edge computing. In [9], Internet of Things (IoT) applications demanding real-time and low latency evaluated and fog computing reduces energy consumption by 40.48% compared to conventional cloud system. Another study compares energy consumption of centralized cloud DC to fog based nano Data Center (nDCs) where flow-based and time-based energy consumption models for shared and non shared network components presented [10]. To implement nano-servers in the fog architecture, assessments and experiments are conducted to generate data for the model.

Applications can benefit from the Fog include smart city and IoT which are percieved as the both modern internet and *Future Internet*. Multi-dimensional data such as audio and video capture from urban sensors can be processed at the fog nodes, and deep-learning models may be trained and perform inferencing to enable real-time decisions such as traffic lights. Autonomus cars and drones can benefit from the Fog with more computational capacity compared to vehicle. MMORPG gaming, 3D environment of HoloLens and Google Glass, robotic surgery with GPGPUs are more possible applications that may well fit to Fog architecture.

Task scheduling in fog computing is a challenging problem due to its heterogeneous, uncertain and dynamic characteristics. The workflow model is the common one to represent applications from various domains on both cloud computing and fog computing paradigms, as well. In this model, a directed acyclic graph is utilized to represent an application where tasks are the nodes and edges are the data dependencies among tasks. Although task scheduling problem is studied for real world applications recently including Industrial IoTs, Smart Factory comprising multiple tasks with dependencies [11–14], most of them address single objective or a small set of weighted objectives. There are a few multi-objective cases, which are mostly inherited from cloud computing and became more strict constraints such as energy, load balancing, reliability and completion time with new challenges such as burden on the global network.

1.1. Thesis Contributions

In this thesis, we model and solve the multi-objective task scheduling problem by exposing prominent features of fog computing with five objectives, which are makespan, energy, cost, reliability, and degree of imbalance. We propose two novel multi-rank based multi-objective scheduling algorithms: the *MOMRank* Algorithm and the *LAMOMRank* Algorithm. Additionally, two task clustering mechanisms are integrated to improve the performance of the algorithm by diminishing costly data transmissions on interconnection topology. An extensive empirical study is conducted with real world applications from Pegasus workflow management system and DeFog benchmark to validate the effectiveness of our algorithms, where they are compared with two well-known meta-heuristics (the NSGA-II and the SPEA2 algorithms) and a widely used method from the literature (the MOHEFT algorithm). Our algorithms, the MOMRank and the LAMOMRank, significantly outperform multi-objective metaheuristic algorithms and they perform comparable with the MOHEFT algorithm on hypervolume (HV) values. The LAMOMRank algorithm addresses the latency demand of real-time applications from the DeFog benchmark by exploring the limits of response latency and round trip time. Our algorithms produce significantly better $\Delta*$ values compared to MOHEFT in less amount of time. To the best of our knowledge, this study is the first attempt to address five objectives, network congestion and latency *simultaneously* as discussed and taxonomically depicted in a recent study [15]. The contributions of the thesis can be listed as follows:

- We propose a hierarchical fog architecture with multi-tiers which are Cloud Tier, Fog Tier and Edge Tier. The fog tier in our model is defined as a tree-based multi layered hierarchical network.
- We propose a task scheduling model targeting five objectives: makespan, energy, cost, reliability, degree of imbalance, then we also highlight average network congestion on clusters, latency for individual real-time tasks.
- We propose a novel multi-rank based multi-objective scheduling algorithm *(MOMRank)* targeting versatility of the problem by utilizing existing multi-objective

evolutionary algorithms and task scheduling algorithms.

- We further extend the MOMRank algorithm to introduce Latency Aware MOMRank algorithm, *LAMOMRank*, to target response latency and delivery time of real-time applications.
- We propose a task clustering mechanism *EdgeMerge* and use an alternate task clustering algorithm *CompMatching* for integrating into algorithms to improve the performance by diminishing costly data transmissions on interconnection topology.
- Performance of the proposed algorithm and extensions are validated on multi-objective metrics to represent comprehensiveness of the performance assessment: *Hypervolume*, $\Delta*$, *Unique Nondominated Front Ratio*

1.2. Outline of the Thesis

The rest of the thesis is structured as follows: Chapter 2 summarizes the related work. In Chapter 3 we propose our multi-tier fog scheme, the system model and problem formulation. Chapter 4 summarizes reference multi-objective algorithms, then we propose our novel algorithm *MOMRank* and an approach for targeting network congestion in Chapter 5. In Chapter 6 we propose latency aware multi-objective task scheduling algorithm *LAMOMRank* and aim to reduce latency of real-time applications. Finally, Chapter 7 concludes the study with future perspectives.

2. LITERATURE SURVEY

Fog computing is a decentralized computing model having many challenging aspects yet to be addressed [6]. Fog computing aims to bring computation and storage near edge of the network using network devices such as gateways, routers. Although task scheduling and load balancing studied abundantly in the literature for cloud computing, fog computing paradigm introduces novel challenges such as decentralization, security/privacy in multi-tier architecture, mobility and agile provisioning. Decentralized architecture of fog computing causes higher level of heterogeneity which increases the complexity of the scheduling problem.

Device or user mobility causes highly dynamic network topology in fog computing environments. Mobility plays crucial role on deciding where to place data and computing of the user applications. Services of users with fast mobility can be placed into higher level resources in the fog hierarchy to reduce number of migrations and to increase network utilization by reducing traffic. Unavailability during migrations are also eliminated by mapping fast moving users application to upper tiers from the QoS aspect. Unlike cloud computing, in fog computing paradigm, there are variety of options for storage and computation of users applications. In some cases, even partial data can be stored at different levels of fog hierarchy. Duplicate or complementary data storage in fog hierarchy will also introduce new security/privacy challenges.

IoT applications can generate vast volume of data which can be used in many different aspects [16]. Fog computing-based face recognition framework to solve security/privacy issues presented in [17]. Fog architecture for big data analysis in smart city concept is investigated in [18]. Road surface condition monitoring for enhancing security in vehicular network by using fog computing architecture was proposed in [19]. In [20], the electroencephalography (EEG) tractor beam game and video surveillance/object tracking application investigated for performance validation of fog computing and proposed simulator. In [21], authors emphasize privacy related issues

of the healthcare related applications. Dependability, privacy concerns and regulations of some applications may restrict sensors data to be offloaded to cloud. In addition, considering the computational limitations of sensor devices at the edge, fog computing suits well to the healthcare applications. Google Glass device based wearable cognitive assistance is presented in [22]. Image capture and sensing capabilities of the Glass is used to perform real-time scene interpretation. However, the processing of captures is done at upper tiers considering battery life and computational limitations of wearable device.

In [23], existing web optimization techniques extended with unique knowledge from fog nodes. Dynamic adaptation of users condition, like network status or device's computing load which are only available at the fog nodes, adapted into web optimization. User's web page rendering performance can be improved beyond webserver of content delivery network. Face recognition method of Picaso [24] used to compare different cloud and fog environments [25]. Authors discuss the mobile application offloading and mobile storage extension in edge intelligence offered by fog computing. In [26] pervasive health monitoring application investigated in fog computing architecture. Authors emphasize that fall is the major source of morbidity and mortality among stroke patients accordingly detecting falls in daily life is crucial. The study investigates new fall detection algorithms and implementation of real-time fall detection algorithm in fog computing architecture.

Fog models are proposed for various scenarios in the literature including smart manufacturing in industrial IoTs [11, 14], connected vehicle networks [27] and smart city [28]. Task and application scheduling in these emerging environments have been studied mostly targeting a selected single objective or the combination of multiple objectives using various aggregation methods into a single one.

Task scheduling and task image placement are jointly investigated in fog computing with software defined embedded systems [29]. Storage and computing devices are disjoined on their underlying model, where task completion time is considered as

the only objective comprising computation time, I/O time and transmission time.

Although a number of multi-objective scheduling algorithms are proposed for fog computing in the literature [30, 31], they mostly emphasize only a subset of objectives covered in Section 3 either jointly or individually, lacking full coverage of all objectives given in this study. Dispersive Stable Task Scheduling (DATS) algorithm is proposed to increase user experience with latency reduction by offloading tasks in heterogeneous fog environments [30]. In their study, the two-phase algorithm is dedicated to first constructing coalition between fog nodes, helper nodes, and Virtual Machines (VMs); then it determines the task allocation vector of each task node. Yang et al. [31] propose a novel delay energy balanced algorithm (DEBTS) using Lyapunov optimization in homogeneous fog model, where the fog model considers mobility of fog devices and peer offloading only between these homogeneous fog nodes. Their algorithm outperforms random-scheduling and the least-busy algorithms yielding lower energy task scheduling with better delay performance.

Prioritized task scheduling in multi-tier fog architecture is tested for a smart factory case scenario by Chekired et al. [11]. Tasks are offloaded to the upper tier fog nodes in the architecture to signify performance of multi-tier fog architecture over flat fog model with task completion and waiting time. Xiao et al. [32] investigate vehicular task offloading and task allocation in Mobile Edge Computing (MEC) systems. The study targets utility maximization via cooperative game theoretic method using gathered real world traffic data. In [33], user allocation in edge computing systems is aimed from service provider's perspective. A game-theoretical approach is used to optimize utilization, user profit and system cost for allocation in decentralized manner. In [34], energy management in fog environments is formulated as an Mixed Integer Linear Programming (MILP) which takes source rate control, load balancing and service replica placement into consideration. A relaxation based heuristic algorithm is used to minimize not only energy usage by the fog nodes, but also targeting increased usage ratio of green energy in the system. Deep reinforcement learning (DRL) is used in [35] to lower operational expenditure as service management and energy consumption, with

emphasis on green and brown energy, for an edge computing environment with varying network topologies.

Nature inspired meta-heuristics are commonly used in the literature for multi-objective domain to overcome the complexity of the problem [12, 13], [36–38]. Task latency and reliability are investigated using an improved version of the NSGA-II algorithm [36]. Artificial ecosystem-based optimization enhanced with slap swarm algorithm is utilized for task scheduling of IoT applications in fog-cloud environments [37]. Performance of makespan and throughput objectives are evaluated individually, where improvements are depicted over other metaheuristics on real world and synthetic workloads. Multi-objective simulated annealing and goal programming are used together in [12] targeting service time, cost, deadline and security objectives. A multi-objective ant colony optimization is proposed with multireplica service placement for scheduling in fog-iot environments [13], where the deployment cost and the service latency are the objectives considered.

In another study, tabu search algorithm is used for load balancing between cloud and fog nodes in [38], which defines the problem as an Integer Linear Programming (ILP) problem. Ye et al. [39] propose multi-access edge computing offloading strategy by modifying the Strength Pareto Evolutionary Algorithm (SPEA2) operators targeting both energy consumption and task execution delay objectives. Bi-objective genetic algorithm (BOGA) is proposed to tackle energy consumption and system reliability for task scheduling in heterogeneous computing systems [40]. In their work, multi-objective differential evolution algorithm and multi-objective Heterogeneous Earliest Finish Time (MOHEFT) algorithm are used as the reference algorithms. The MOHEFT [41], a list scheduling based algorithm, is one of the most commonly used multi-objective task scheduling algorithm in cloud-edge computing, which is also the baseline algorithm in our experimental study.

The OpenFog Consortium was founded for standardization of fog computing architecture. Main objective is to make standards for IoT devices to connect and work in

harmony securely with different edge and cloud services. The OpenFog consortium has released a white paper on fog computing [42], presenting the Open Fog architecture in consortium's perspective. The OpenFog consortium announces OpenFog Reference Architecture in 2017 [43]. This architecture is the prime act for creating open fog computing architecture. The OpenFog Consortium provides guidelines and standards that is coordinated with standards institutions such as IEEE.

We summarize existing fog scheduling models from literature in Table 2.1. Fog scheduling models in the literature usually have limitations of addressing multiple issues simultaneously. In our fog scheduling framework, we address as many objectives as possible from list of open issues presented in studies that propose and promote fog computing paradigm [6], [42]. To the best of our knowledge, our fog scheduling model is one of the most comprehensive one in the literature as taxonomically depicted in recent study [15].

Fog benchmarks are introduced in the literature to measure efficiency and effectiveness of fog computing systems. These benchmarks are used to assess various aspects of the fog architectures like throughput, latency, resource utilization, cost and energy. In [20], with IFogSim simulator, real world applications are presented to be used on simulated fog architectures: electroencephalography (EEG) tractor beam game and video surveillance/object tracking application. In another fog computing benchmark, DeFog [44], various applications are presented and enabled to be distributed on cloud-edge continuum. Applications can be executed on cloud-only, edge-only and cloud-edge distributed schemes. Various metrics of underlying processing node and applications are available. We utilize DeFog benchmark to include applications in our workloads as bag of tasks beside the Pegasus workflows.

Table 2.1. Properties of existing fog scheduling models.

	Energy	Cost	Makespan (Computation Latency)	Reliability (Fault Tolerance)	Response Latency	Workload balancing	Multi-tier architecture in fog layer	Network Congestion
[45] Sarkar et. al.	Yes	Yes	Yes	No	No	No	No	No
[29] Zeng et. al.	No	No	Yes	No	No	No	No	No
[30] Liu et. al.	No	No	Yes	No	No	No	No	No
[31] Yang et. al.	Yes	No	Yes	No	Yes	No	No	Yes
[46] Mishra et. al.	Yes	Yes	Yes	No	No	No	No	No
[47] Pham et. al.	No	Yes	Yes	No	No	No	Yes	No
[48] Wu et. al.	Yes	No	No	No	No	No	No	No
[49] Niu et. al.	No	No	No	No	No	No	No	No
[36] Sun et. al.	No	No	Yes	Yes	No	No	No	No
Our model	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

3. SYSTEM MODEL AND PROBLEM FORMULATION

3.1. System Model

In our fog architecture, the system has three tiers: *Cloud Tier*, *Fog Tier* and *Things Tier* (see Figure 3.1). The *Fog Tier* contains hierarchical deployment of clusters where each cluster is represented by a tree with varying depth levels. Tasks in a workflow can be distributed on the submitter edge node, fog nodes or cloud nodes. Workloads can not be allocated on neighbouring edge nodes which may violate privacy. It causes performance degradation for limited performance capacity of edge nodes as well. Fog computing nodes are attached physically to the local network devices: gateways, switches and routers. In our model, we assume resources are abundant on *Cloud Tier* and getting limited towards edge of the network. In *Things Tier*, smart homes, mobile devices, smart city infrastructures, wearable computing devices are expected to be receiving services from fog clusters.

3.2. Problem Formulation

In this study, we formulate our fog scheduling model with five objectives which are minimizing *makespan, energy, cost, degree of imbalance* and maximizing *reliability*. The workloads submitted to system are represented with directed acylic graphs (DAGs) where there are dependencies and transmission requirements between tasks. We assume that DAGs in the following model and our experimental study are single exit task and single entry task graphs. If a DAG has multiple entry and/or exit nodes, we extend it with pseudo entry and/or exit tasks with 0 workload and 0 communication costs to its predecessors and/or successors.

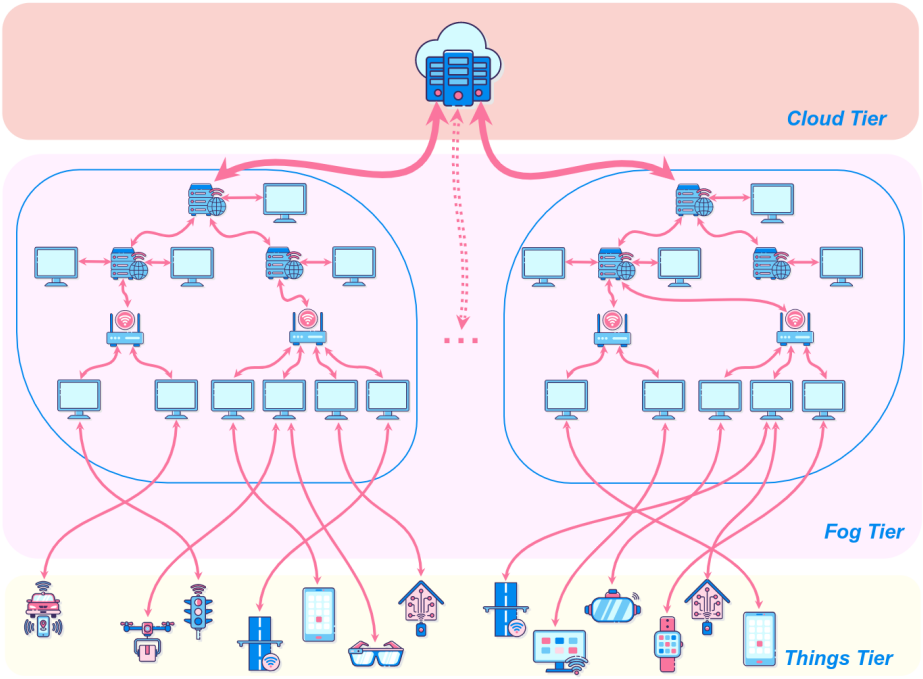


Figure 3.1. Architecture of fog system model.

3.2.1. Makespan

The first objective is the minimization of the makespan which is measured by the time between submission of the workload and finish of the workload. Makespan objective can be expressed as

$$min.\ \ (FT(t_{exit})),\ \ where\ \ t_{exit} \in DAG, \tag{3.1}$$

where $FT(t_{exit})$ is the finish time of exit task t_{exit} of the given DAG.

Each task must be allocated to exactly one of the cloud, fog or edge computing nodes as in

$$s.t.\ \ \sum_{j=1}^{N} y_{mj} = 1, \tag{3.2}$$

where m is the task number and j is the processing node id, which is either cloud, fog

or edge node. Correspondingly, y_{mj} allocation variable is set to 1 if task m allocated to node j, 0 otherwise. We assume that task executions are non-preemptive in our scheduling model.

In following equation, we define finish time (FT) of the task n as

$$FT(t_n) = \begin{cases} \left[max(FT(t_m) + \frac{d_{mn}}{\tau_{ji}}, ready_j) + \frac{w_n}{\rho_j} \right] & \text{, if } y_{ni} = 1 \text{ and } y_{mj} = 1 \\ \text{where } m \in pred(n) & \\ \frac{w_n}{\rho_j} & \text{, if } y_{ni} = 1 \\ \text{where } pred(n) = \emptyset & \end{cases} , \quad (3.3)$$

which is the maximum of machine ready time ($ready_j$) and arrival of predecessor tasks' dependent data. Workload of task n is represented by w_n; and the data transmission requirement between task m and task n is represented by d_{mn} where task m is a predecessor of task n (m $\in$ pred(n)). The terms y_{ni} and y_{mj} are allocation variables to processing nodes from cloud tier, fog tier and edge tier. The processing capacity of node j is given as ρ_j and the transmission capacity of link between nodes i and j in each tier is given as τ_{ij}. The term $ready_j$ represents the ready time of the node j which is assumed always 0 for cloud allocations indicating processing nodes are available abundantly.

3.2.2. Energy

The second objective is the minimization of the energy consumption, written as

$$min\ \zeta = \zeta^{process} + \zeta^{trans}, \quad (3.4)$$

where task processing of the computing nodes and data transmission between computing nodes are operations demanding energy in the system.

Total processing energy consumption of the computing nodes can be expressed as

$$\zeta^{process} = \sum_{m=1}^{T} \sum_{j=1}^{N} y_{mj} \times \epsilon_j \times \frac{w_m}{\rho_j}, \tag{3.5}$$

where y_{mj} represents allocation decision (Equation 3.2) and ϵ_j is the energy consumption per time unit for node j. The term w_m/ρ_j represents the time spent for processing task m on node j.

Following equation accumulates transmission energy consumption for each data dependency between tasks

$$\begin{aligned} \zeta^{trans} &= \sum_{m=1}^{T} \sum_{\substack{n=1 \\ n \neq m}}^{T} \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{d_{mn}}{\tau_{ij}} \times \epsilon_{ij}^{trans}, \\ s.t.\ y_{mi} &= 1, y_{nj} = 1 \\ m &\in pred(n) \end{aligned} \tag{3.6}$$

where d_{mn} is the data dependency between task m and task n; ϵ_{ij}^{trans} is the data transmission energy consumption per time unit between node i and node j. In this model, the idle time energy consumption for processing nodes is neglected.

3.2.3. Cost

The third objective is the cost minimization. The total cost spent for allocation to cloud, fog and edge tiers calculated as follows

$$min \;\; C = \sum_{m=1}^{T} \sum_{j=1}^{N} y_{mj} \times c_j \times \frac{w_m}{\rho_j}, \tag{3.7}$$

c_j represents cost per unit time of processing node j, where y_{mj} is a allocation decision (Equation 3.2).

3.2.4. Reliability

In our model, we measure the reliability of processing and data transmission using predefined failure probability of each processing node and inter-node transmission link, respectively. We target to maximize system reliability, which is computed with the product of overall reliability of processing and overall reliability of data transmission expressed as

$$max \;\; R = \prod_{m \in T} R^{process}(t_m) \times \prod_{d \in D} R^{trans}(d_{mn}), \tag{3.8}$$

where all m tasks d edges of the DAG is considered.

Weibull distribution is considered since it is the closest distribution for modeling resource failures [50]. Probability of failure for each machine is expressed as

$$p(x) = \frac{\beta x^{\beta-1}}{\eta^{\beta}} e^{-(\frac{x}{\eta})^{\beta}}, \tag{3.9}$$

where η and β are scale and shape parameters of the Weibull distribution, respectively. When we integrate the failure probability of processing node for the time spend on processing and transmission, then the failure rate of each task is derived as

$$\begin{aligned} R^{process}(t_m) &= 1 - \int_0^{\frac{w_m}{\rho_j}} p_j(x)dx = e^{(-\frac{\frac{w_m}{\rho_j}}{\eta})^{\beta}}, \\ s.t.\ y_{mj} &= 1, \end{aligned} \tag{3.10}$$

where the term $\frac{w_m}{\rho_j}$ represents amount of time spent for executing task m on node j.

The reliability of inter task data transmission expressed as

$$\begin{aligned} R^{trans}(d_{mn}) &= 1 - \int_0^{\frac{d_{mn}}{\tau_{ij}}} p_{ij}(x)dx = e^{-(\frac{\frac{d_{mn}}{\tau_{ij}}}{\eta})^{\beta}}, \\ s.t.\ y_{mi} &= 1,\ y_{nj} = 1\ , \\ m &\in pred(n) \end{aligned} \tag{3.11}$$

where the term p_{ij} represents failure rate of link from fog node i to fog node j at any time instant x.

3.2.5. Degree of Imbalance

One of the main motivations of technology shift from cloud computing to fog computing is the transmission of big data produced by the connected devices to the cloud. Big data transmission causes the global network to slowdown. This is also valid for the fog tier where low degree of imbalance in the task allocation in fog cluster can cause overloading of fog nodes closer to the edge of the network. Balancing workload inside of the fog cluster can prevent choking the network closer to the edge and it improves the response time for the new job arrivals.

Degree of imbalance for fog node j expressed as

$$DI_j = \sum_{m=1}^{T}\sum_{j=1}^{N} y_{mj} \times \frac{w_m}{\rho_j}. \tag{3.12}$$

The objective is to minimize overall degree of imbalance DI, which is calculated by

$$min\ \ DI = \frac{DI_{max} - DI_{min}}{DI_{avg}}, \tag{3.13}$$

where DI_{max}, DI_{avg} and DI_{min} are the maximum, average and minimum degree of imbalance in fog nodes respectively.

4. MULTI OBJECTIVE TASK SCHEDULING IN FOG COMPUTING

In a multi-objective optimization problem (MOP), there are two or more objectives where at least two of them may be in conflict with one another. For a MOP, Pareto-optimal set (POS) is the set of solutions that are not dominated by any other solutions in decision space. Similarly, Pareto-optimal front (POF) is the set of solutions that is determined in objective space. Formally, A MOP is stated as

$$min \;\; F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x})), \tag{4.1}$$

where $f_1(\vec{x})$ is the first objective, m is the number of objectives. In the following section, we present the details of the MOHEFT algorithm [41], which is the baseline algorithm for our experimental study. Two widely used multi-objective evolutionary algorithms from literature, the NSGA-II Algorithm [51] and the SPEA2 algorithm [52], are considered in the empirical evaluations, as well.

4.1. MOHEFT Algorithm

Multi-objective Heterogeneous Earliest-Finish-Time (MOHEFT) [41] algorithm is a Pareto-based list scheduling algorithm extended from the classical task scheduling algorithm for heterogeneous resources, the HEFT [53]. Figure 4.1 summarizes outline of the MOHEFT Main idea of the MOHEFT is to track all objectives over all resource assignments and greedily select partial schedules for the next assignment. It evaluates m x K partial schedules at each iteration and selects K tradeoffs for the next iteration where m is the number of resources. Tasks are sorted according to *Urank*, where *Urank* of a task is the maximum cumulative distance of the task to the end of the workflow by considering the workloads and the data transmissions of the workflow. Motivation of ranking is to give precedence into tasks that are on the critical path of the workflow. Crowding distance is used to select partial schedules into next iteration of ranked tasks.

Require: W=(A,D), A=$\bigcup_{i=1}^{n} A_i$ ▷ Workflow application
Require: R=$\bigcup_{i=1}^{m} R_i$ ▷ Set of resources
Require: K ▷ Number of tradeoff solutions
1: **function** MOHEFT(W,R,K)
2: Rank ← URANK(A) ▷ Order the tasks according to Urank
3: **for** k ← 1, K **do** ▷ Create K empty workflow schedules
4: $S_k \leftarrow \emptyset$
5: **end for**
6: **for** i ← 1, n **do** ▷ Iterate over the ranked tasks
7: $S' \leftarrow \emptyset$
8: **for** j ← 1, m **do** ▷ Iterate over all resources
9: **for** k ← 1, K **do** ▷ Iterate over all tradeoff schedules
10: $S' \leftarrow S' \cup \{S_k \cup (Rank_i, R_j)\}$ ▷ Add new mapping to all intermediate schedules
11: **end for**
12: **end for**
13: $S' \leftarrow SORTCROWDDIST(S', K)$ ▷ Sort according to crowding distance
14: $S' \leftarrow FIRST(S', K)$ ▷ Choose K schedules with highest crowding distance
15: **end for**
16: **return** S
17: **end function**

Figure 4.1. MOHEFT algorithm.

$l = |S|$ ▷ Number of solutions in S
for $i \leftarrow 1, l$
1: $S[i]_{distance} = 0$ ▷ İnitialize distances
2: **end for**
for each objective m
3: S=sort(S,m) ▷ Sort for each objective
4: $S[1]_{distance} = S[l]_{distance} = \infty$ ▷ Make sure boundary points are always selected
5: **for** $i \leftarrow 2, (l-1)$ ▷ For all other points
6: $S[i]_{distance} = S[i]_{distance} + (S[i+1].m - S[i-1].m)/(f_m^{max} - f_m^{min})$

Figure 4.2. crowding-distance-assignment(S)

4.2. NSGA-II

NSGA-II algorithm [51], one of the most commonly used multi-objective meta-heuristic in the literature including Fog Scheduling [36]. Algorithm 4.3 summarizes outline of the NSGA-II algorithm. Initially random population will be generated and recombination, selection mechanisms will be called for fixed number of generations. *fast-non-dominated-sort()* function classifies solutions according to their domination relation into the *frontiers*($\mathcal{F}_1, \mathcal{F}_2$ .., e.g.). In each frontier set, solutions will not dominate each other. Next generation of the population will be selected from frontier sets, until the population size reached. Provided that, population size exceeded with addition of last frontier, last frontier is truncated with respect to their crowding distance. Crowding distance calculation is given in Figure 4.2. Crowding distance of a solution is calculated by distance to its nearest solutions respect to each objective. Boundary points for each objective assigned with value of inf to be selected into next iteration.

1: **for** t $\leftarrow$ 1, T **do** ▷ Number of generation
2: $R_t = P_t \cup Q_t$ ▷ Combine parent and offspring population
3: $\mathcal{F}$ =fast-non-dominated-sort(R_t) ▷ $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, ...)$, all nondominated fronts of R_t
4: $P_{t+1} = \emptyset$ and $i = 1$
5: until $|P_{t+1}| + |\mathcal{F}_i| \leq N$ ▷ Until the parent population is filled
6: crowding-distance-assignment($\mathcal{F}_i$) ▷ Calculate crowding-distance in $\mathcal{F}_i$
7: $P_{t+1} = P_{t+1} \cup \mathcal{F}_i$ ▷ Include ith nondominated front in the parent pop
8: $i = i + 1$ ▷ Check the next front for inclusion
9: Sort($\mathcal{F}_i, \prec_n$) ▷ Sort in descending order using $\prec_n$
10: $P_{t+1} = P_{t+1} \cup \mathcal{F}_i[1 : (N - |P_{t+1}|)]$ ▷ Choose the first $(N - |P_{t+1}|)$ elements of $\mathcal{F}_i$
11: Q_{t+1} = make-new-pop(P_{t+1}) ▷ Use selection, crossover and mutation to create a new population Q_{t+1}
12: **end for**

Figure 4.3. NSGA-II algorithm.

4.3. SPEA2

Strenght Pareto Evolutionary Algorithm 2 (SPEA2) [52], is an multi-objective evolutionary algorithm an improved version of its predecessor. Extension of SPEA2, over SPEA is a fitness calculation which considers both dominance relation and density of the candidate solutions in the search space. Algorithm 4.4 summarizes outline of the SPEA2 algorithm. Population and archive jointly recombined for a predefined number of generations. Best individuals are stored into archive and transfered into next generation. If nondominated solution set is larger than the archive size, $\overline{N}$, portion of archive is truncated depending on the fitness value. Contrarily, if number of individuals in the archive is less than the archive size, dominated individuals depending on the density value added into the next population.

Input: N (population size)
$\overline{N}$ (archive size)
T (maximum number of generations)
Output: A (nondominated set)

function SPEA2($N, \overline{N}, T$)
initialize-Population P_0 ▷ Initialize feasible random schedules
$\overline{P_0} = \emptyset$ ▷ Initialize empty archive set
for $t \leftarrow$ 1,T ▷ T is number of generation
    calculate-Fitness(P_t) ▷ Calculate fitness of population
    calculate-Fitness($\overline{P_t}$) ▷ Calculate fitness of archive
    $\overline{P}_{t+1} \leftarrow$ nondominatedSet($P_t \cup \overline{P_t}$) ▷ Copy all nondominated individuals from $\overline{P}_t$ and P_t to archive
    if $|\overline{P}_{t+1}| > \overline{N}$
        truncate($\overline{P}_{t+1}$) ▷ Truncate portion of archive exceeds the archive size
    if $|\overline{P}_{t+1}| < \overline{N}$
        fill-dominated-solutions($\overline{P}_{t+1}$) ▷ Fill archive with dominated solutions if archive size is not reached
    $P_{t+1} \leftarrow$ make-new-pop($\overline{P}_t$)
end for
end function

Figure 4.4. SPEA2 algorithm.

Fitness assignment in SPEA2 population can be calculated considering domination and distance to other individuals in the population.*Strength value* $S(i)$ can be calculated by

$$S(i) = |\{j | j \in P_t \cup \overline{P}_t \wedge i \succ j\}|, \tag{4.2}$$

where $|.|$ refers to cardinality of the total population and $i \succ j$ means individual i, dominates individual j. The raw fitness value $R(i)$ based on the S value calculated as

$$R(i) = \sum_{j \in P_t \cup \overline{P}_t, j > i} S(j). \tag{4.3}$$

We can summarize raw fitness as accumulated strength values of dominators in overall population. Minimum fitness individual better in this representation. Raw fitness value is insufficient considering a population where dominance relation is uncommon. Thus distance relation is injected into fitness calculation to improve search space exploration. Distance to k-th member is considered for each individual in the sorted population and archive list. k depends on the population and archive size ($k = \sqrt{N + \overline{N}}$). Since the fitness value is a minimization, density $D(i)$, inverse distance relation of individual i can be calculated as

$$D(i) = \frac{1}{\sigma_i^k + 2}. \tag{4.4}$$

In the denominator, 2 is added to make density greater than zero and less than 1. Finally fitness value $F(i)$ can be calculated as

$$F(i) = R(i) + D(i). \tag{4.5}$$

5. A NEW MULTI ASPECT HEURISTIC: MOMRANK

In this thesis, we present a new task scheduling algorithm, called *Multi-Objective Multi Rank (MOMRank) Algorithm*, by utilizing multiple ranking schemes with maintaining a fixed size set for tradeoff solutions. We use three rank schemes to sort tasks which are upward rank, workload rank and computation to communication rank. Upward rank [41] gives priority to the tasks according to their distance to the exit node considering workload. In workload rank, we start from the entry node and without violating the task precedence we incrementally rank tasks with their workload. In computation to communication ratio, we sort the tasks according to ratio of their workload over incoming and outgoing data requirements, again by considering task precedence. Tradeoff solutions are non-dominated set of solutions among different objectives.

Algorithm 5.1 outlines the MOMRank algorithm where P is the total number of task ranking schemes, K expresses number of output schedules of the workflow, and R is the number of resources. The rank of the tasks are set in line 3; and then lines 8-10 are for assigning tasks to the resources for each rank set. In line 11 and 12, solutions are sorted and K/P solutions are preserved where K is the predetermined number of tradeoff solutions and P is the number of ranks. Finally, line 14 returns the K tradeoff solutions. For each task rank strategy, we maintain isolated solution sets, because of different task allocation at each iteration. The MOMRank algorithm also evaluates m x K schedules at each iteration, in total. Thus, the complexity of the MOMRank algorithm would be $O(P.n.m.\frac{K}{P})$. The number of tradeoff solutions (K) and the number of task ranking algorithms would be significantly less than the number of tasks and resources, m and n. Therefore, the time complexity of the MOMRank algorithm is $O(n.m)$.

Require: W=(A,D), A=$\bigcup_{i=1}^{n} A_i$ ▷ Workflow application
Require: R=$\bigcup_{i=1}^{m} R_i$ ▷ Set of resources
Require: K, Number of tradeoff solutions, P, Number of ranks.
1: **function** MOMRank(W,R,K)
2: **for** p ← 1, P **do**
3: Rank ← $RANK_p(W)$ ▷ Order the tasks according to ranking scheme
4: **for** k ← 1, K/P **do** ▷ Create K/P empty workflow schedules
5: $S_{pk} \leftarrow \emptyset$
6: **for** i ← 1, n **do** ▷ Iterate over ranked tasks
7: $S'_p \leftarrow \emptyset$
8: **for** j ← 1, m **do** ▷ Iterate over all resources
9: **for** k ← 1, K/P **do** ▷ Iterate over all tradeoff schedules
10: $S'_p \leftarrow S'_p \cup \{S_{pk} \cup (Rank_i, R_j)\}$ ▷ Add new mapping to all intermediate schedules
11: $S'_p \leftarrow SORT\_CROWD\_DIST(S'_p, K/P)$ ▷ Sort according to crowding distance
12: $S'_p \leftarrow FIRST(S'_p, K/P)$ ▷ Choose K/P schedules with highest crowding distance
13: S=$\bigcup_{p=1}^{P} S_p$
14: **return** S

Figure 5.1. MOMRank algorithm.

5.1. Targeting Fog Network Congestion

In the era of *IoT* and *Big Data*, network congestion becomes a vital aspect of the new network architectures. As part of our MOMRank algorithm, we aim to lower network congestion caused by data movement of the workflows. We expect that eliminating large data movements by allocating source and destination tasks to the same physical node will relieve network burden on the fog cluster. Therefore, we utilize a low complexity DAG clustering algorithm (the *CompMatching* algorithm given in [54]). In their work, a DAG clustering approach is proposed in three phases: coarsening, initial partitioning and refinement. In the first phase, the tasks are clustered using their topological value, which avoids cyclicity to yield coarser DAGs. The second phase computes the initial partitioning; and a refinement algorithm is used to get better partitioning

of the clusters at the last phase. In this thesis, we utilize coarsening phase due to its low complexity. For the *MOMRank* algorithm and the reference algorithms, we adapt coarsening phase where iteratively tasks are clustered until number of vertices is lower than a threshold. Figure 5.2 outlines coarsening algorithm *CompMatching* which computes matching of the vertexes in [54]. The *CompMatching* algorithm marks edges iteratively, if topological order difference is less than 1 (line 8,12) and the destination vertex does not have multiple sources (line 8) which would cause cyclicity.

```
    Input: Directed Graph G = (V,E), a traversal order of the vertices in V, a priority on edges
    Result: A feasible matching M of G
 1: match ← ∅
 2: top ← CompTopLevels(G)
 3: for u ∈ V do mark[u]← false
 4: for u ∈ V following the traversal order in input do
 5:       if mark[u] then continue
 6:       for v ∈ Pred[u] ∪ Succ[u] following given priorities on edges do
 7:            if mark[v] then continue
 8:            if (top[u] ≠top[v]-1) and (|Pred[v]|≠ 1) and (|Succ[u]|≠ 1) then continue
 9:            if v ∈ Pred[u] then
10:                 M ← M ∪ {(v, u)}
11:                 for w ∈ Succ[v] do
12:                      if top[v] = top[w] − 1 then
13:                           mark[v]← false
14:           else
15:                 M ← M ∪ {(u, v)}
16:                 for w ∈ Succ[u] do
17:                      if top[u] = top[w] − 1 then
18:                           mark[w]← false
19:           mark[u]←mark[v]← false
20: return M
```

Figure 5.2. CompMatching algorithm.

In this study, we propose an alternative low complexity approach to the *CompMatching* algorithm for DAG clustering, the *EdgeMerge* strategy where the main idea is to eliminate large data movements in fog network. Figure 5.3 outlines steps of our

task clustering strategy. In this strategy, we sort edges with respect to their weights in line 3 which represent data dependency and possible data transmission on a cluster. Then, we select top E edges in line 4 and allocate both source and destination tasks of these edges to the same physical fog node in lines 5-7. Algorithms make a decision during the task allocation phase, if a link between the task and predecessor is decided to be eliminated, then the task is allocated to the same physical computing node. If an algorithm decides to change allocation of one of these bundled tasks at any step, it will allocate corresponding bundled tasks to the same new physical node.

```
   Input: Workflow W= (T,D), tasks in T, edges as task dependencies in D, number of edges to
   be merged in E
   Result: Clusters of tasks
1: function (W)
2: Cluster[] ← T                                  ▷ Initially each task in distinct cluster
3: D* ← Sort(D)
4: for e ← 1, E do
5:       src ← D*[e].source
6:       dst ← D*[e].destination
7:       Cluster[src]= Cluster[src] ∪ Cluster[dst]
8: return Cluster[]
```

Figure 5.3. EdgeMerge algorithm.

Figure 5.4 depicts the flow chart of our framework in detail for multi-objective task scheduling in fog environments. The framework requires the physical fog cluster networks and workflows as inputs. If any repair on workflows are necessary, pseudo entry and/or exit tasks are added. Then, the task clustering algorithm is selected if any integration is going to be applied. Then, the results of all algorithms including our proposed one are used to assess the approximate pareto front. Finally, multi-objective performance assessment is presented using multi-objective metrics and approximate pareto front.

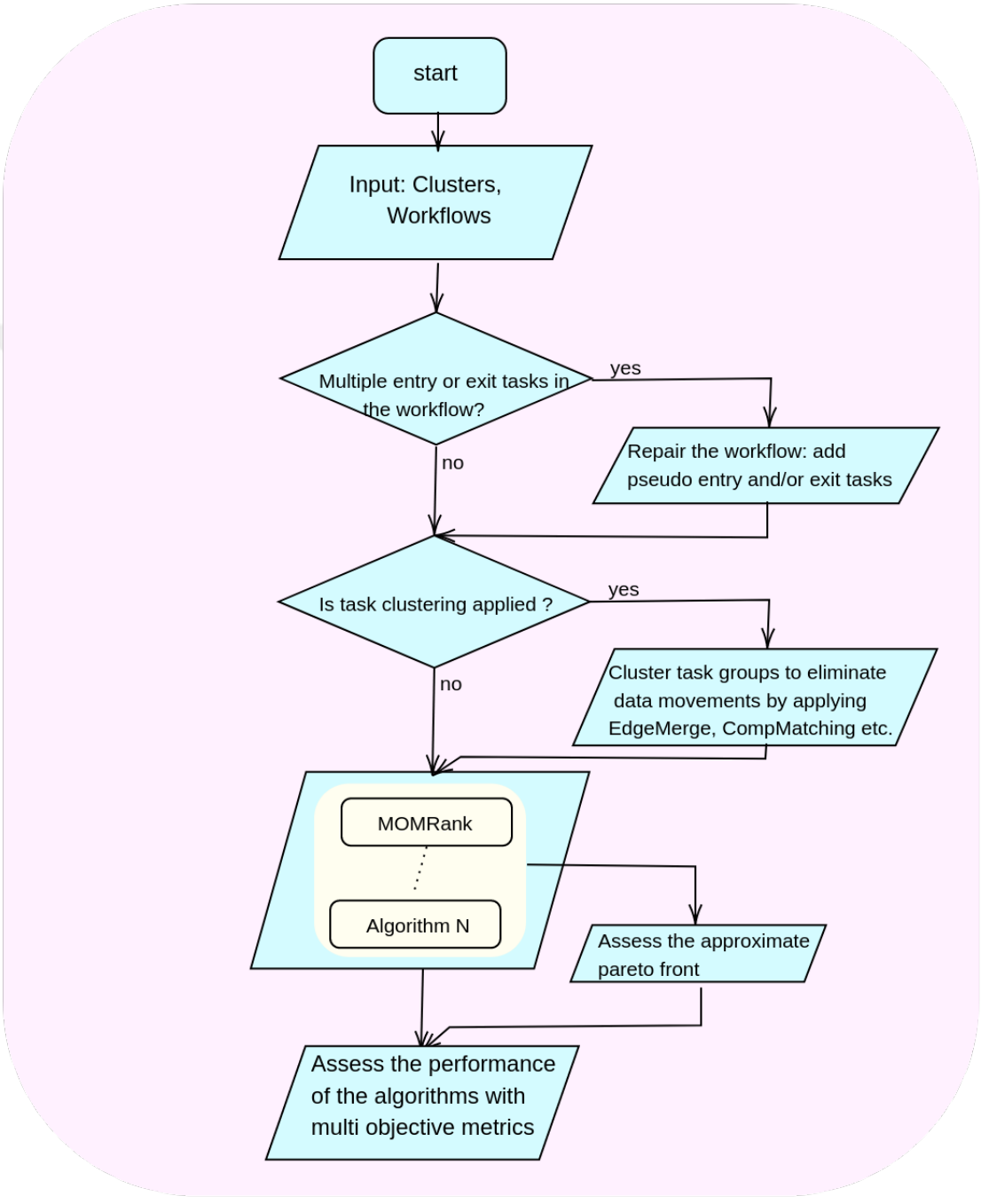


Figure 5.4. Flow chart of our multi-objective scheduling framework for fog environments.

5.2. Experimental Setup

The performance of the MOMRank algorithm is validated with the MOHEFT, NSGA-II and SPEA2 algorithms, where all algorithms in the empirical study are coded in C programming language and run on the same device. In order to provide fair comparisons, same operators and parameter values are considered for all algorithms, unless otherwise stated. Single-point crossover is used for recombination; and two mutation operations are used with equal probability: a new node assignment to a random task, and reposition of a task in the allocation order. The mutation probability is set to 0.3; and the merged edge count E is set as 10. Any task in the workflow can be allocated into cloud, a fog node or the submitter edge node as explained in Section 3.

5.3. Pegasus Workflow and Amazon EC2

We evaluate the algorithms by utilizing workflows from Pegasus Workflow Management System [55] with various topologies from different domains. There are five different workflows which are CyberShake, Montage, Inspiral, Epigenomics and Sipht, where Figure 5.7 symbolically represents their topological structures. We consider instances of Amazon Elastic Compute Cloud (Amazon EC2), which provides variety of resizable capacity for CPU, memory storage and networking in the cloud, tailored for different application profiles. Table 5.1 presents the computing capacity (ρ), network bandwidth (τ), cost (C) and reliability (β) of the instances. Costs of these instances vary on depending on the host operating system. We select RHEL pricing due to scaling of costs per time unit. For example in Linux instances, cost will scale in exact amount of CU which would eliminate an objective in our model. Units are insignificant in most of the cases where each metric will be normalized into range of [0,1] for multi-objective metric calculations (see section 5.5).

We use computing units on each instance (CU) as computational capacity, i.e., it defines how many independent computing units available on corresponding instance. Thus, we calculate task execution which is given by a single node execution time in

the Pegasus workflows.Data size units are given in workflow executions in bytes per second; and Table 5.1 illustrates network bandwidths in Gbps.

Table 5.1. Amazon EC2 parameters used in our experimental study.

	Computing				Networking					
					Down			Up		
	ρ	ϵ	C	β	τ	ϵ	β	τ	ϵ	β
m5.large	2	1.00	0.175	1.6	10	1.00	2.2	3	1.00	2.2
m5.xlarge	4	1.80	0.29	1.8	10	1.00	2.2	3	1.00	2.2
m5.2xlarge	8	3.24	0.59	2	10	1.00	2.2	3	1.00	2.2
m5.4xlarge	16	5.83	1.05	2.2	10	1.00	2.2	3	1.00	2.2
m5.8xlarge	32	10.50	1.97	2.4	12	1.32	2.4	3.6	1.32	2.4
m5.12xlarge	48	18.90	2.89	2.6	20	2.42	2.6	6	2.42	2.6
m5.16xlarge	64	34.01	3.81	2.8	25	3.33	2.8	7.5	3.33	2.8
m5.24xlarge	96	61.22	5.65	3	NA	NA	NA	NA	NA	NA

We scale energy consumption by 1.8 which is not available to end users. We also select variety of β values for Weibull distribution for the reliability objective. Main motivation is to generate parameter rates with varying scales, otherwise objectives scaled with same rates would limit the search space.

Conversion of β values into reliability using Weibull distribution illustrated in Figure 5.5. Our motivation is to limit minimum reliability into some threshold. We select largest task on the workflow and lowest computing capacity node (edge node: *m5.large*), to determine τ value used for Weibull distribution. We multiply execution time of longest task on *m5.large*, which is illustrated as a point where distributions coincide.

5.4. Deployment Schemes

In experimental study, two physical fog network in tree topology is implemented using Amazon EC2 computing nodes (see the clusters in Figure 5.6(a) and 5.6(b)).

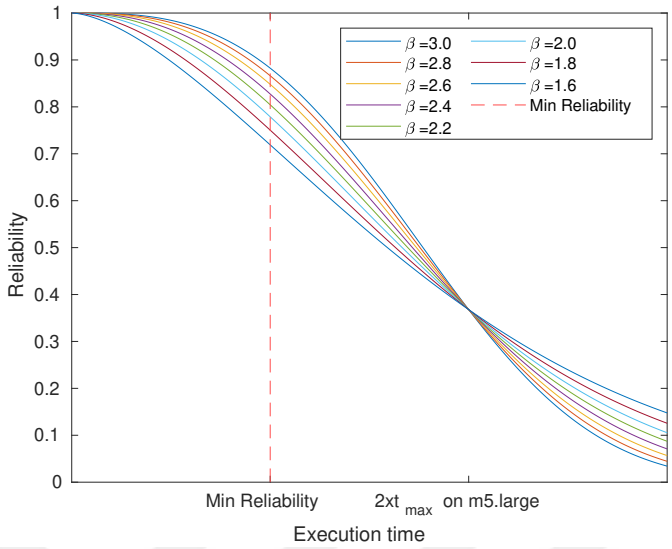


Figure 5.5. Minimum possible reliability value for computing node.

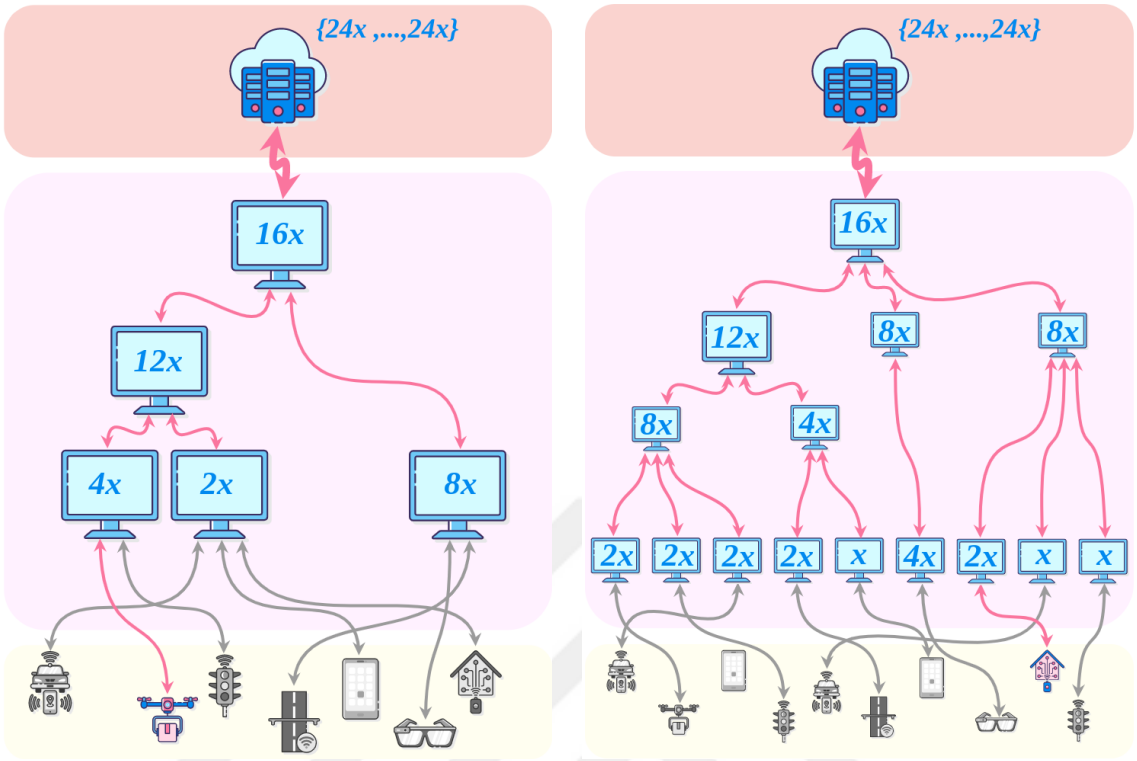
We use *m5.24xlarge* instances on cloud side where abundant number of instances are available. Edge nodes will have instance of *m5.large* where only submitter edge node can be used as a computing alternative to the cloud and fog in our model. In *Fog Tiers*, we use variety of instances from table 5.1 ranging from *m5.xlarge* to *m5.16xlarge*.

5.5. Metrics for Multi-Objective Optimization

In order to assess performance of algorithms for multi-objective optimization problems, there are a number of metrics presented in the literature [56]. In our empirical evaluations, we select three widely used metrics in the literature, which are: Unique Nondominated Front Ratio (UNFR), Δ^*, and the Hypervolume (HV)respectively.

Since the true pareto front is unknown for the experiments, we combine all non dominated solutions from evaluated solvers as the approximate pareto front, P. Set of solutions produced by algorithms will be referred as S for following metric definitions.

Unique Nondominated Front Ratio (UNFR): metric indicates portion of non-dominated solutions in the solution set. One of the issues about the commonly used ver-



(a) Cluster 1. (b) Cluster 2.

Figure 5.6. Deployment schemes.

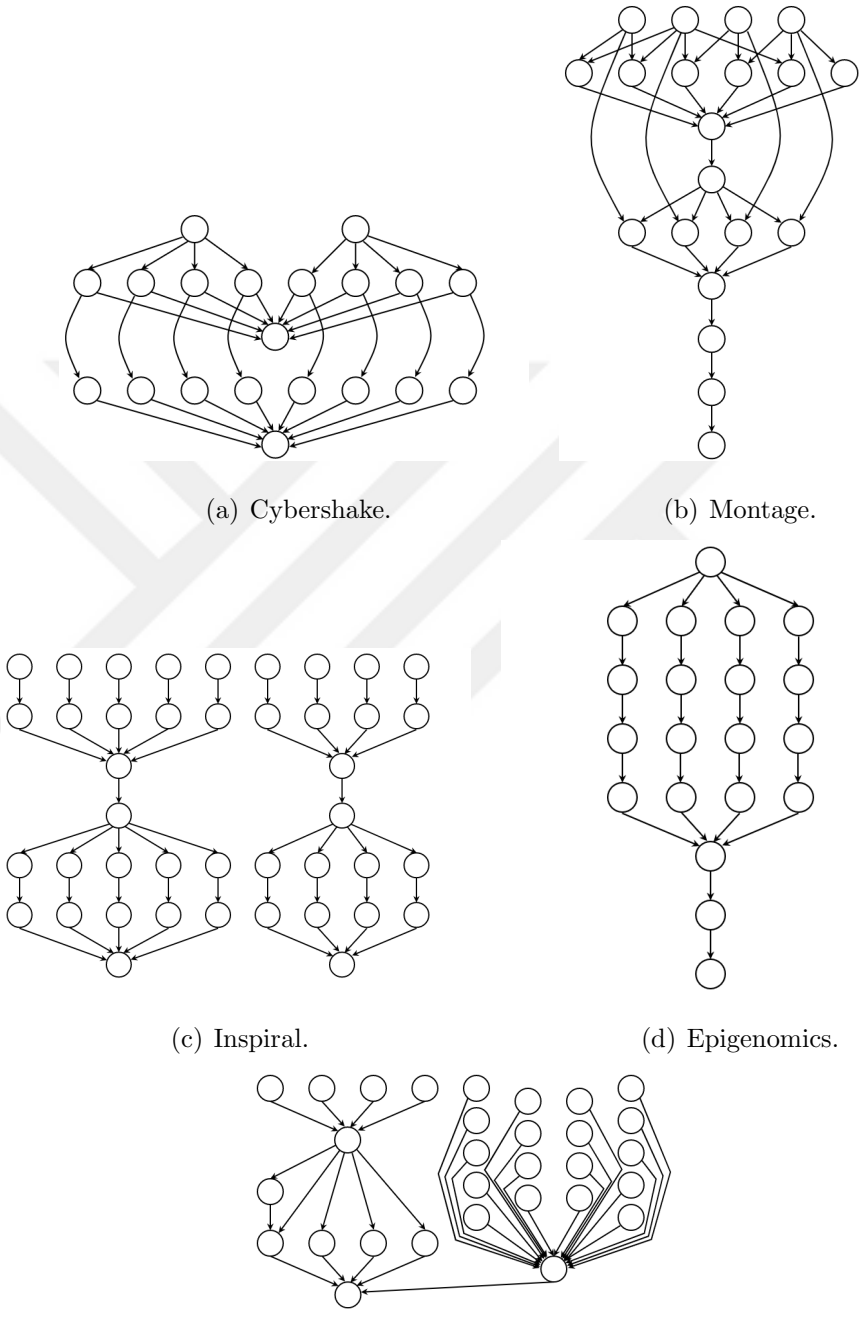
sion of this metric is that some multi-objective solvers may output duplicate solutions where duplicates would not dominate each other. Uniqueness in the non-dominated set is injected into this metric to address this challenge [56], expressed as follows

$$UNFR(S) = \frac{|s \in S_{unf} | \nexists r \in R_{unf} : r \prec s|}{|R_{unf}|}. \tag{5.1}$$

In this equation, R_{unf} represents reference unique nondominate front. We select unique S as R_{unf} by discarding duplicate solutions for the initial experimental study; therefore, the values of this metric will be in [0,1].

Δ^* metric measures the spread of the solutions in S expressed as

$$\Delta^*(S,P) = \frac{\sum_{k=1}^{m} d(\vec{e}_k, S) + \sum_{i=1}^{|S|} |d_i - \overline{d}|}{\sum_{k=1}^{m} d(\vec{e}_k, S) + (|S|)\overline{d}}, \tag{5.2}$$



(a) Cybershake.

(b) Montage.

(c) Inspiral.

(d) Epigenomics.

(e) Sipht.

Figure 5.7. Pegasus workflow.

where $d(\vec{e}_k, S) = \min_{\vec{s} \in S} ||F(\vec{e}_k) - F(\vec{s})||$ and $\vec{e}_k \in P$ is the extreme solutions on k-th objective. $\overline{d}$ is the average distance in the set S.

Hypervolume (HV) is one of the most commonly used metrics in literature despite its high complexity [56]. Hypervolume measures the search space volume bounded by the known pareto front. Different objectives would have different value range resulting discrepancy by emphasizing effect of higher range in the unnormalized HV calculation. Thus, we normalize each objective value to the known maximum and minimum values of corresponding metric dimension. In our empirical evaluations, we consider a low complexity implementation of HV metric presented in the literature [57].

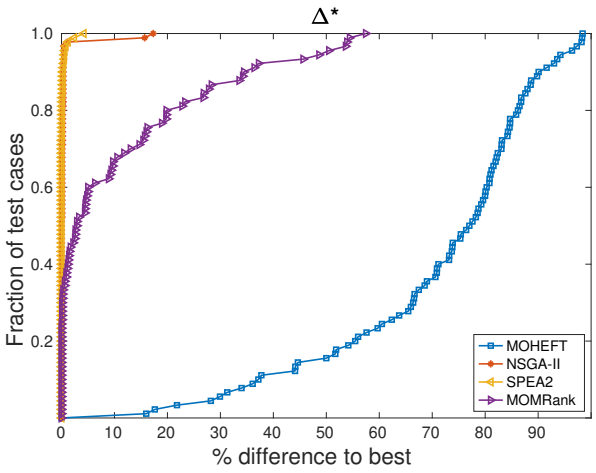
We consider average network congestion on fog clusters,*Average Network Congestion(ANC)*, in addition to these multi-objective metrics, for fog network congestion extension of our MOMRank algorithm (see Section 5.1). Average network congestion (ANC) on a fog cluster can be calculated as

$$ANC = \frac{\sum_{l=1}^{L} busy\_time(link_l)}{L}, \tag{5.3}$$

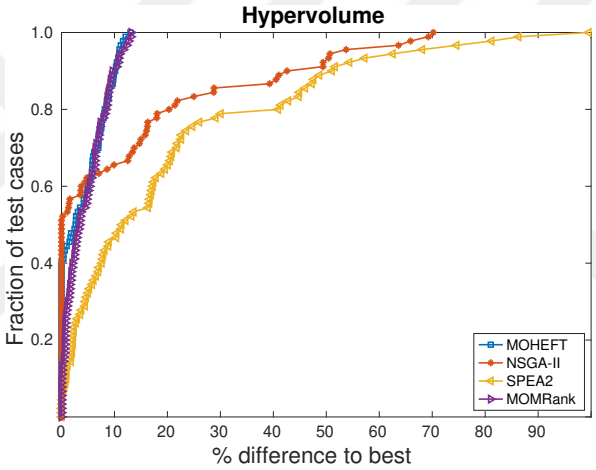
where $busy\_time(link_l)$ is the time spent for data transmission on physical link $link_l$ in a cluster; and L is the number of physical link in a fog cluster.

5.6. Experimental Results

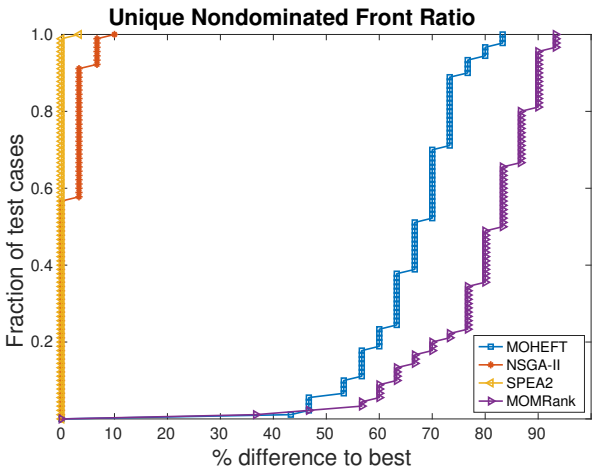
The comparison study of the algorithms are performed by using 15 real workloads, which are derived by using 5 workflow structures of the Pegasus system with 3 instances per case by considering up to 100 tasks. The number of instances is increased by scaling computation to communication ratio (CCR) of workflows taken from median processing node and median network parameters from Table 5.1. Thus, using 5 different CCR values from the {0.2,0.5,1,5,10} and the default CCR value given in [55], we conduct experimental study with 90 workflows. Metaheuristic algorithms are executed in the literature with fixed number of generations. On our preliminary experiments,



(a) $\Delta*$.

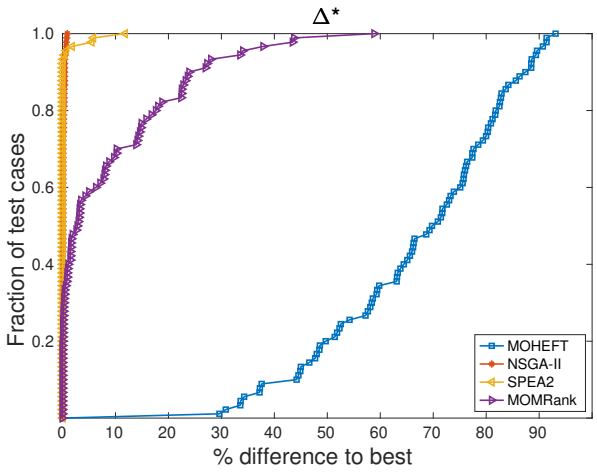


(b) HV.

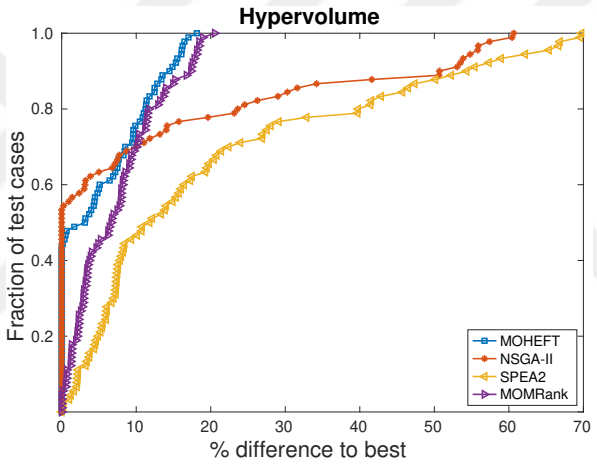


(c) UNFR.

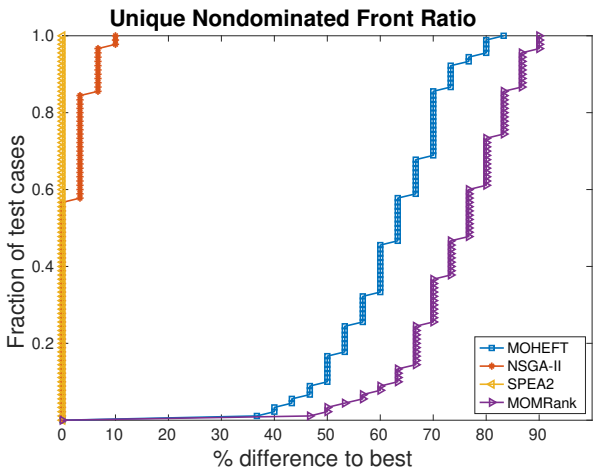
Figure 5.8. Algorithm performances on cluster 1 for multi-objective metrics $\Delta*$, *HV* and *UNFR*.



(a) $\Delta*$.



(b) HV.



(c) UNFR.

Figure 5.9. Algorithm performances on cluster 2 for multi-objective metrics $\Delta*$, *HV* and *UNFR*.

Table 5.2. Performance comparison for multi-objective metrics in selected set of workflows with number of tasks is 100.

		Hypervolume				$\Delta*$				UNFR			
		MOHEFT	MOMRank	NSGA-II	SPEA2	MOHEFT	MOMRank	NSGA-II	SPEA2	MOHEFT	MOMRank	NSGA-II	SPEA2
Cluster 1	Montage100	**0.81**	0.80	0.19	0.26	0.17	0.53	**1.00**	**1.00**	0.47	0.17	**0.97**	0.93
	CyberShake100	**0.84**	0.82	0.67	0.56	0.38	**1.00**	**1.00**	**1.00**	0.27	0.23	0.97	**1.00**
	Epigenomics100	**0.82**	**0.82**	0.61	0.59	0.21	0.35	0.99	**1.00**	0.37	0.27	**1.00**	**1.00**
	Inspiral100	**0.81**	**0.81**	0.44	0.39	0.44	0.99	**1.00**	0.99	0.33	0.37	0.97	**1.00**
	Sipht100	0.83	0.82	**0.84**	0.81	0.11	0.49	0.98	**0.99**	0.23	0.17	**1.00**	**1.00**
Cluster 2	Montage100	**0.74**	0.72	0.35	0.34	0.22	0.86	**1.00**	**1.00**	0.47	0.20	**1.00**	**1.00**
	CyberShake100	**0.76**	0.75	0.59	0.62	0.17	0.59	**1.00**	**1.00**	0.30	0.23	**1.00**	**1.00**
	Epigenomics100	**0.76**	0.74	0.56	0.45	0.21	0.21	0.99	**1.00**	0.47	0.30	**1.00**	**1.00**
	Inspiral100	**0.73**	**0.73**	0.26	0.21	0.16	0.45	**1.00**	**1.00**	0.37	0.20	**1.00**	**1.00**
	Sipht100	0.79	0.74	**0.85**	0.71	0.34	0.70	**1.00**	0.98	0.27	0.20	**1.00**	**1.00**

we observe that metaheuristic-based techniques are up to 200 times slower than the MOHEFT and the *MOMRank* algorithms when the iteration count is fixed to 10000. Thus, we limit the execution time of SPEA2 and NSGA-II, where we stop generation count when those algorithms reaches the execution time of the MOHEFT algorithm. Due to nature of the MOMRank algorithm which splits the solution set into three and sorts them at each iteration where MOHEFT sorts triple size, MOMRank has lower execution times compared to MOHEFT up to 50% for all test cases. For the rest of the thesis, execution times are equal for the reference algorithms unless otherwise stated.

We present three multi-objective metrics for 100 tasks of 5 workflows from Pegasus benchmark In Table 5.2.The *MOMRank* algorithm reaches the performance of MOHEFT with respect to HV metric with at most 2% degradation. However, on spread metric of $\Delta*$, it improves up to 4 times. The *MOMRank* algorithm outperforms the NSGA-II algorithm for 8 out of 10 instances over to 4 times in HV metric; and it outperforms the SPEA2 for all instances up to 3.5 times. On the other hand, metaheuristic algorithms provide better spread ($\Delta*$) values and unique nondominated solution ratios compared to MOHEFT and *MOMRank*.

We illustrate rest of the experimental results with performance profiles [58] which helps to compare performance of algorithms in aggregated fashion over all problem instances. Figures 5.8, 5.9 depict performance of the four algorithms (without considering enhancements of our algorithm on network congestion) on cluster 1 and cluster 2 in three metrics, Δ^* , HV and UNFR. On this experiment (Figure 5.8, 5.9) x-axis represents each algorithm's performance respect to performance of the best algorithm for corresponding instance in percentage and y-axis represents the fraction of the instances. Thus, the closer to the y-axis is the better algorithm for given metric. Minimum area between performance curves and the y-axis indicates the best performance for the corresponding metric. Partial lines on the y-axis indicates the algorithm reached the best values for the designated fraction of test cases.

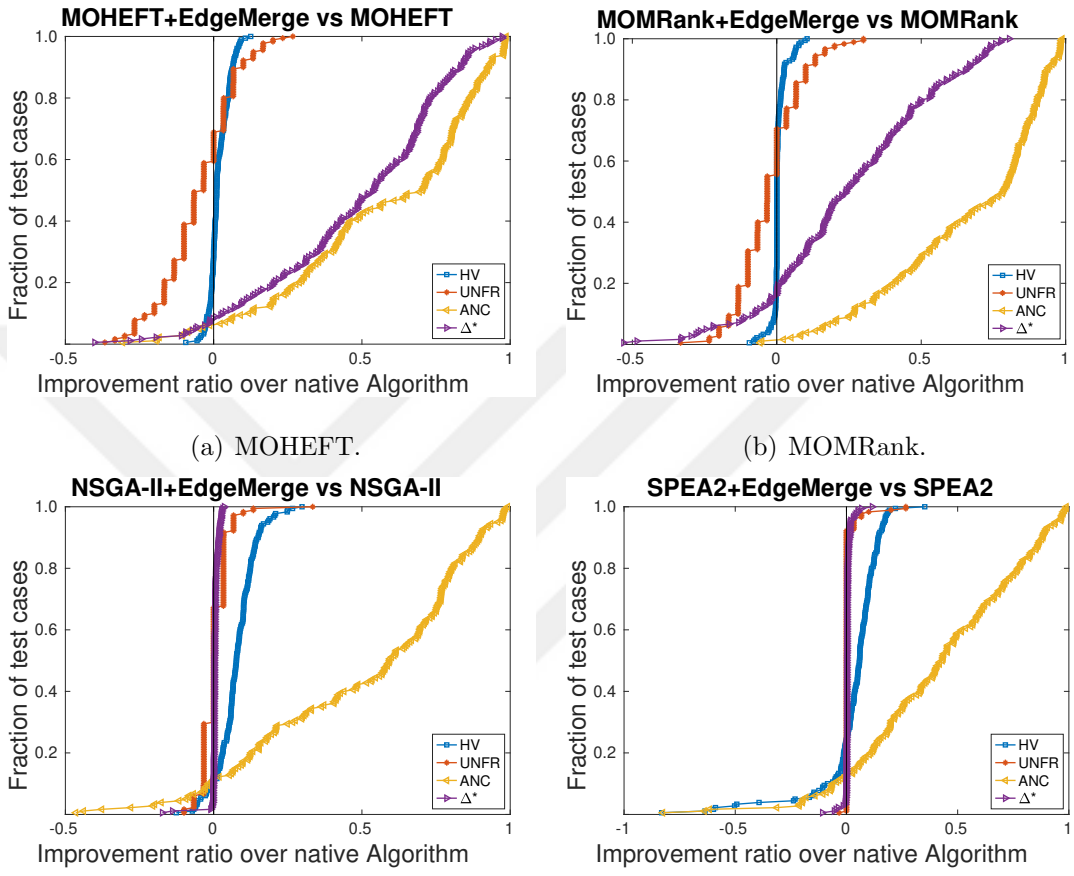
5.6.1. Performance Analysis on Multi-objective Metrics

The *MOMRank* algorithm reaches to the best spread value (Δ^*) for 33% of the test instances which is due to the multiple task ranking approaches that cause diversity (Figure 5.8(a)) with respect to MOHEFT. On the othe hand, the MOHEFT algorithm does not reach the best spread for any of the instances; it is 15% away from the best Δ^* for its best case. The NSGA-II and SPEA2 algorithms reach the best results in spread values (Δ^*), for almost every instances validating search space exploration feature of the meta-heuristics. For *Hypervolume* values (given in Figure 5.8(b)), *MOMRank* and *MOHEFT* algorithms present almost the same best performance with 13% decline on worst case yielding smaller hypervolume values for cluster 1. However, *MOMRank* starts to degrade when the search space is widened on cluster 2 for almost %60 of instances compared to *MOHEFT*(given in Figure 5.9(b)). SPEA2 algorithm has worst performance with respect to the HV values where the NSGA-II has the best value for 52% of instances. When UNFR metric is considered, the MOMRank algorithm generates less number of unique solutions compared to MOHEFT (see Figure 5.8(c)). The SPEA2 algorithm outperforms other algorithms yielding most number of unique solutions for almost every instances.

5.6.2. Performance Analysis with Network Congestion Aware Extensions

In another experimental setup, we measure the performance effect of eliminating data transmission by mapping tasks to same physical node. In Section 5.1, we present two approaches to extend our algorithm with the objective of eliminating high data transmission on physical network, which are: the *EdgeMerge* and the *Comp-Matching* algorithms. We separately illustrate *Average Network Congestion* (ANC) value further extending the objective set which already comprises 5 different objectives(Section 3). ANC is calculated by the average time spent on data transmission over every physical network link (Section 3.2, Equation 5.3). With the ANC metric, we ascribe higher prominence to network congestion for upcoming future internet beyond objectives in Section III.B. Figure 5.10 depicts impact of DAG clustering approaches

on the performances of the algorithms. Each algorithm is compared with its native version considering three metrics and the ANC value for each of 90 instances on 2 clusters.



(a) MOHEFT.

(b) MOMRank.

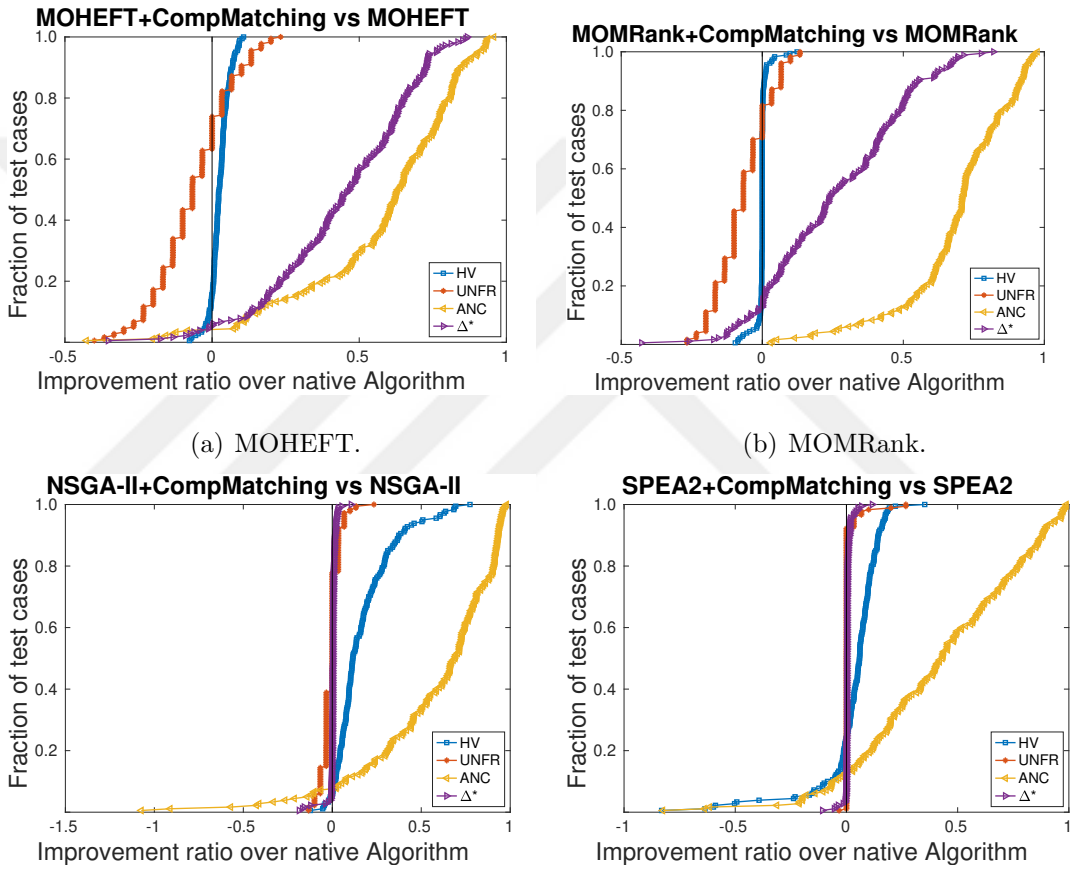
(c) NSGAII.

(d) SPEA2.

Figure 5.10. Impact of DAG clustering extensions for multi-objective metrics and average network congestion on algorithms with EdgeMerge extension.

The algorithms yield worse results for some multi-objective metrics like UNFR (Figure 5.10(a),5.10(b),5.11(a),5.11(b)). It is because of the fact that limiting the search space by merging tasks may result more duplicate solutions in the MOHEFT and *MOMRank* solution sets. Evolutionary metaheuristics are more robust in terms of exploration of search space which are less affected by this limitation. We observe that for almost 60% of the instances, network extensions lead to worse UNFR values. However, while there are slight or no improvements on the HV values, the Δ^* values significantly improve for the MOHEFT and the *MOMRank* algorithms. Extensions of

the *CompMatching* and the *EdgeMerge* lowers the ANC values for every algorithm for more than 95% of instances. For the meta-heuristic techniques, we observe comparatively slight changes on multi-objective metrics compared to the MOHEFT and the *MOMRank* algorithms. The HV values are improved for the SPEA2 and the NSGA algorithms for 60% and 80% of the instances on both extensions, respectively (Figure 5.10(d),5.11(d)).



(a) MOHEFT.

(b) MOMRank.

(c) NSGAII.

(d) SPEA2.

Figure 5.11. Impact of DAG clustering extensions for multi-objective metrics and average network congestion on algorithms with CompMatching extension.

Table 5.3. Statistical analysis of algorithms based on pair wise comparisons with respect to HV metric.

	MOHEFT	MOHEFT+CompMatching	MOHEFT+EdgeMerge	MOMRank	MOMRank+CompMatching	MOMRank+EdgeMerge	NSGAII	NSGAII+CompMatching	NSGAII+EdgeMerge	SPEA2	SPEA2+CompMatching	SPEA2+EdgeMerge
MOHEFT	*NA*	s-	s-	s+	s+	s+	s+	s-	+	s+	s+	s+
MOHEFT+CompMatching	s+	*NA*	+	s+	s+	s+	s+	s-	+	s+	+	s+
MOHEFT+EdgeMerge	s+	-	*NA*	s+	s+	s+	s+	s-	+	s+	+	s+
MOMRank	s-	s-	s-	*NA*	-	-	s+	s-	+	s+	s-	s+
MOMRank+CompMatching	s-	s-	s-	+	*NA*	-	s+	s-	+	s+	s-	s+
MOMRank+EdgeMerge	s-	s-	s-	+	+	*NA*	s+	s-	+	s+	s+	s+
NSGAII	s-	s-	s-	s-	s-	s-	*NA*	s-	s-	+	s-	-
NSGAII+CompMatching	s+	s+	s+	s+	s+	s+	s+	*NA*	s+	s+	s+	s+
NSGAII+EdgeMerge	-	-	-	-	-	-	s+	s-	*NA*	s+	-	s+
SPEA2	s-	s-	s-	s-	s-	s-	-	s-	s-	*NA*	s-	s-
SPEA2+CompMatching	s-	-	-	s+	s+	s-	s+	s-	+	s+	*NA*	s+
SPEA2+EdgeMerge	s-	s-	s-	s-	s-	s-	+	s-	s-	s+	s-	*NA*

Last experiment targets to illustrate the statistical analysis of algorithm performance based on pairwise comparisons. We consider the HV metric for this experiment. We conduct Wilcoxon rank sum test for all 180 instances (90 workflows on 2 clusters) to statistically validate performance of the algorithms, where all HV values are normalized. Each result given in each cell of Table 5.3 compares the performance of the row algorithm with the performance of the column algorithm. Cells are marked with 's+', 's-', '+' and '-' to show that the corresponding row algorithm significantly better than, significantly worse than, better than or worse than corresponding column algorithm for the HV metric, respectively. Table 5.3 highlights that performance improvements for our task clustering extensions, *CompMatching* and *EdgeMerge*, on Figure 5.10 are statistically significant. We can also conclude that NSGAII extended with the proposed *CompMatching* clustering scheme is the best performing algorithm.

6. EXTENDING MOMRANK BY TARGETING FOG NETWORK LATENCY

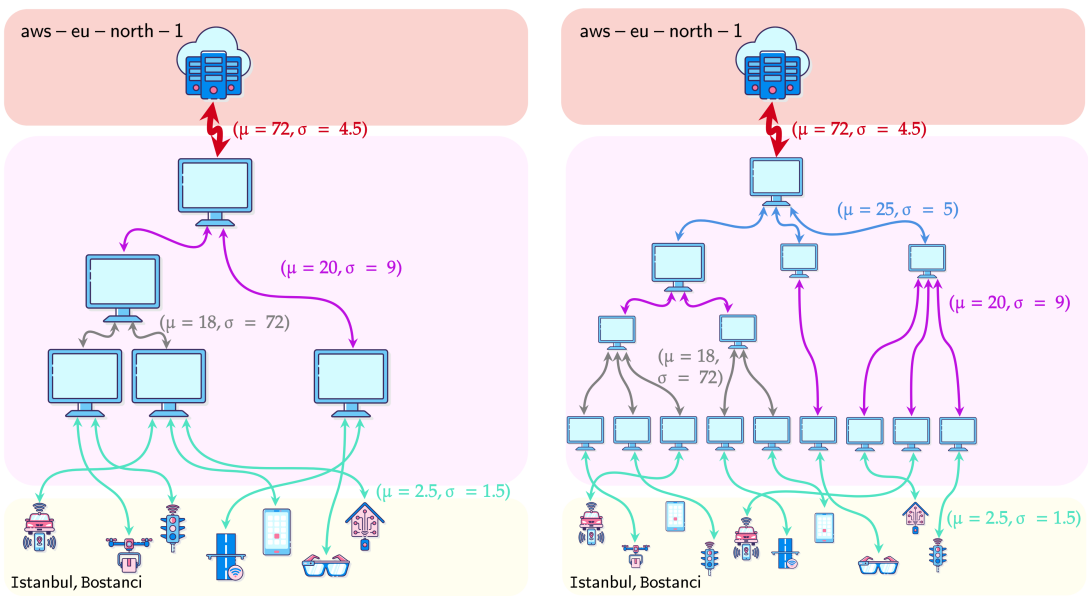
Low latency demand of real world applications is one of the driving motivation for fog computing. Applications deployed to the conventional cloud computing instances causing delays and low quality service for real time requirements due to passing through many hop points for a data transmission on global network [6]. Latency constrained applications must be favored to be allocated towards the edge of the network in the edge to cloud continuum to fulfill real time, low latency demand. In this chapter, we extend the MOMRank algorithm given in Chapter 5 and propose a *Latency Aware Multi-objective Multi Rank Algorithm* (LAMOMRank) to favor latency demanding applications on extended workload set.

We augment our workloads with applications demanding real time service requirements to be deployed on our two fog clusters. Pegasus workflows comprise diverse set of dependent tasks which can be easily partitioned to be distributed over fog clusters. Although Pegasus benchmark has various properties that are easily scalable and deployable for distributed computing, it lacks real time features where single task delivery time or host response latency is not an objective or constraint. The DeFog [44] benchmark (See Section 6.3) is selected for extending Pegasus workflows as a bag of tasks paradigm due to its nature representing real world applications with various characteristics. We assess the performance of algorithms from Chapter 5 and LAMOMRank on modified workload instances for multiobjective metrics and two new latency metrics for DeFog benchmark applications : *Weighted Average Response Latency* (WARSP) and *Average Round Trip Time* (ARTT).

The WARSP metric can be a significant service requirement as live applications being deployed and running on a node where user interacts with application in real time. On the other hand, the ARTT metric can be a representative QoS measurement for applications that require run and deliver the output to the user.

6.1. Targeting Fog Network Latency

Network response latency values are collected using the *traceroute* and the *mtr* commands on linux development PC. These tools can be used to diagnose the routers information and statistics as the network packets traveled to the given destination IP. We sent Internet Control Message Protocol (ICMP) packets to Amazon Web Services eu-north-1 (Stockholm, Europe) and collected statistics of response latencies through every hop points. Response latency of some hop points gradually selected to be used in our experimental study.Figure 6.1 depicts experimentally used response latency values for each link with mean and standard deviation values in *ms*. It can be easily asserted that applications deployed towards the edge of the cluster will have significantly lower network response latency compared to a cloud node or hierarchically distant fog node, independent of their computational capacity. This allocation approach is expected to improve user experience for real time applications. In our experimental study, if a DeFog application is deployed to a fog or a cloud computing node, total response latency will be accumulated on the path from the submitter node to the target computing node.



(a) Cluster 1 response latency.

(b) Cluster 2 response latency.

Figure 6.1. Response latency assignment for computing nodes.

6.2. Latency Aware MOMRank

MOMRank algorithm, proposed in Chapter 5, aims to tackle miscellaneous challenges of the heterogeneous fog scheduling problem taking into account objectives given in Section 3.2. Robustness of the MOMRank algorithm is the result of its' diversified, multiple task ranking schemes. We further modify the MOMRank algorithm to put latency demand on emphasis for the DeFog benchmark applications.

Require: W=(A,D), A=$\bigcup_{i=1}^{n} A_i$ ▷ Workflow application
Require: R=$\bigcup_{i=1}^{m} R_i$ ▷ Set of resources
Require: K, Number of tradeoff solutions, P, Number of ranks.
function LAMOMRank(W,R,K)
  for p ← 1, P **do**
    Rank ← $RANK_p(W)$ ▷ Order the tasks according to ranking scheme
    for k ← 1, K/P **do** ▷ Create K/P empty workflow schedules
      $S_{pk} \leftarrow \emptyset$
    for i ← 1, n **do** ▷ Iterate over ranked tasks
      $S'_p \leftarrow \emptyset$
      for j ← 1, m **do** ▷ Iterate over all resources
        for k ← 1, K/P **do** ▷ Iterate over all tradeoff schedules
          if $LATENCY\_CONSTRAINED(A_i)$
            if $ON\_PATH\_TO\_CLOUD(R_j)$
              $S'_p \leftarrow S'_p \cup \{S_{pk} \cup (Rank_i, R_j)\}$
            else
              $S'_p \leftarrow S'_p \cup \{S_{pk} \cup (Rank_i, R_{submitter(A_i)})\}$
          else
            $S'_p \leftarrow S'_p \cup \{S_{pk} \cup (Rank_i, R_j)\}$
      $S'_p \leftarrow SORT\_CROWD\_DIST(S'_p, K/P)$ ▷ Sort according to crowding distance
      $S'_p \leftarrow FIRST(S'_p, K/P)$ ▷ Choose K/P schedules with highest crowding distance
  S=$\bigcup_{p=1}^{P} S_p$
return S

Figure 6.2. LAMOMRank algorithm.

Figure 6.2 outlines the LAMOMRank algorithm which vary from MOMRank algorithm at lines 10-16. We first divert the allocation decision of the DeFog applications, which are assumed to be in real-time applications, compared to Pegasus tasks in line 10. If a resource node R_j is not on the path from submitter edge node to the cloud, the LAMOMRank algorithm does not allow allocation (Lines 11-14). Duplicate partial schedule will be added to the solution set S'_p with submitter edge node allocation in line 14. To summarize, LAMOMRank algorithm aims to explore latency front (WARSP vs ARTT) by allowing real time applications, in our experimental case DeFog tasks, to be allocated only on the exact path from submitter edge node to the cloud. Intuitively, it is expected that allocations diverted from this path will increase response latency (WARSP) while not much gaining on the computation power (ARTT) for the DeFog applications.

6.3. DeFog Benchmark and Bag of Tasks

We extend our workload instances by the DeFog [44] benchmark which is proposed to address open challenges in Fog environments. The benchmark provides a total of six applications for Fog Computing paradigm and provides three deployment modes: *cloud-only*, *edge-only* and *cloud-edge (Fog)*. It also has a prominent feature of presenting the comprehensive set of metrics from both application perspective and platform perspective. We select three applications from DeFog benchmark to further extend our instances: *YOLOv3*, *PocketSphinx* and *Aeneas*.

YOLOv3 (The You Only Look Once) represents a real-time object detection system rooted in deep learning. It employs a singular neural network to analyze the entire image. The image is broken down into distinct regions, with the system estimating bounding boxes and probabilities for each of these regions. These bounding boxes are then weighted based on the predicted probabilities. *PocketSphinx* is an open-source, speaker-independent, large vocabulary speech recognition engine designed for continuous speech recognition. Its functionality involves the conversion of a supplied audio file in the .wav format into text, using a pre-trained acoustic model that is specifically

tailored to ascertain both the source and target language for the speech-to-text conversion process. The *Aeneas* tool carries out the automatic synchronization of text and audio segments. The process of creating synchronization maps for a collection of text excerpts and an audio file that contains the corresponding text is commonly known as forced alignment. Aeneas, in this context, identifies and establishes the mapping between the relevant audio segments for each provided text asset.

The DeFog applications are provided as docker containers that can be easily deployed and run on any edge, fog or cloud nodes. We deploy and run these three applications in cloud-only mode on Amazon EC2 m5.large instance as in Pegasus workflows execution. Upload image size in MB, execution time in seconds and download results in MB metrics are collected and these metrics are fed into workload space to create bag of tasks as illustrated on Figure 6.3.

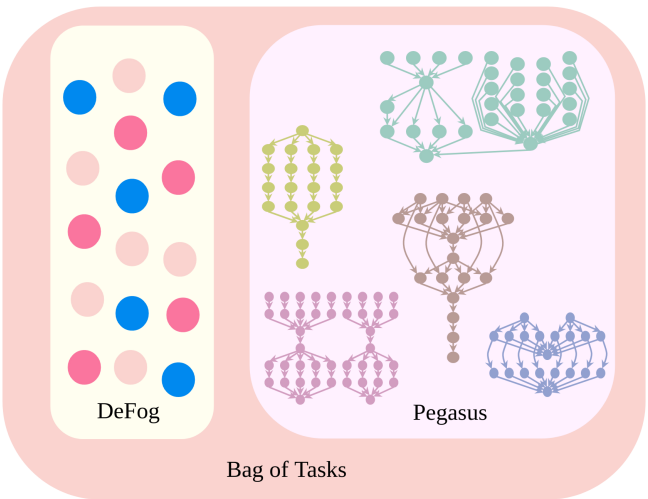


Figure 6.3. *Bag of Tasks* populated with Pegasus workflows and DeFog benchmark.

We scale the DeFog applications workload and data (container image to be uploaded, results to be downloaded) to create balance with Pegasus workflow instances both in terms of total workload and number of tasks. We assume that workloads are submitted before the scheduling causing non-dependent parallel workloads (the DeFog vs the Pegasus) populate each instance. A repair operation on problem instances are required as in the case of Pegasus workflows with multiple entry and/or multiple exit

tasks. This is due to the Bag of Tasks workload model enabling parallel executions where there is no dependency between Pegasus and DeFog. Figure 6.4 depicts an example repair operation for system model consistency with injection of pseudo entry/exit tasks and data dependencies between non-dependent tasks.

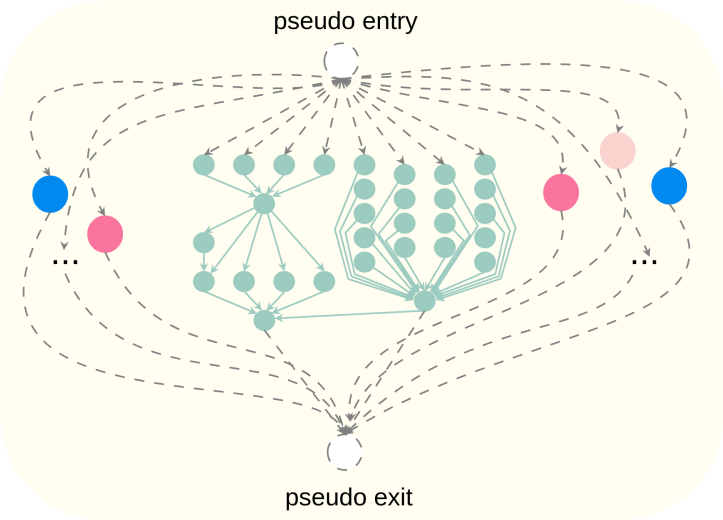


Figure 6.4. An example repair operation for workload model consistency.

We preserve the number of instances as in Chapter 5 but each instance is approximately doubled in workload with DeFog applications. Populating instances with set of independent tasks will relax search space by reducing average dependency between two tasks and enabling more parallel executions for schedulers.

6.4. Response Latency and Computation Latency

In this chapter, we assess the performance of the algorithms in terms of two latency metrics: *Weighted Average Response Time* (WARSP) and *Average Round Trip Time* (ARTT) along with multiobjective metrics presented in Section 5.5. WARSP metric is the average of response latencies of DeFog applications weighted on workload size. Response latencies used for each node in our experimental studies for two clusters

illustrated in Figure 6.1. WARSP can be calculated as

$$\begin{aligned} WARSP = \sum_{m=1}^{T}\sum_{j=1}^{N} y_{mj} \times \Big(\sum_{l=1}^{L} \frac{response\_latency_l}{w_m}\Big), \\ s.t.\ \ l \in Path(R_{submitter(m)}, R_j), \\ m \in \{DeFog\} \end{aligned} \tag{6.1}$$

where $response\_latency_l$ is the response latency value for link l. $Path(R_{submitter(m)}, R_j)$ returns the set of links that are on the path from submitter edge node ($R_{submitter}$) to allocated resource (R_j).

ARTT metric is used to measure the delivery time of computational tasks offloaded to cloud or fog node. It measures the average time spend between start of the task image upload to the computing node and finish time of the output data download for DeFog applications. ARTT can be given as

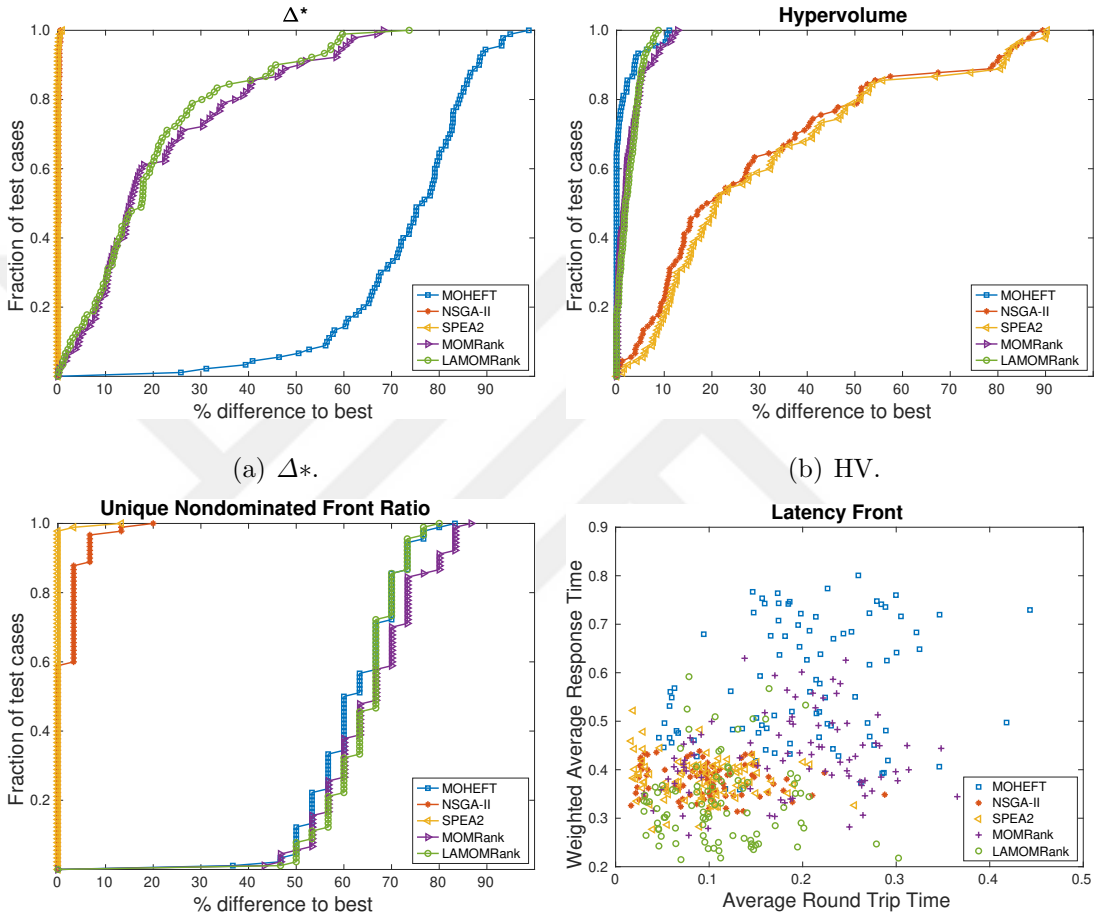
$$\begin{aligned} RTT(m) = FT(out_{\downarrow_m}) - ST(img_{\uparrow_m}), \\ ARTT = \sum_{m=1}^{T} \frac{RTT(m)}{T}, \\ s.t.\ m \in \{DeFog\} \end{aligned} \tag{6.2}$$

where $RTT()$ is the round trip time for the single task, $out_{\downarrow_m}$ is download of output data and $img_{\uparrow_m}$ is upload of the image data. Then, ARTT is calculated as average over DeFog tasks.

6.5. Experimental Results

We use experimental set up explained in Section 5.2 unless otherwise stated. Some of the experiments given in Chapter 5 are repeated due to change in the characteristics of the workload instances. Number of tasks that can be run on parallel are drastically increased with injection of DeFog applications. Extended algorithms with task clustering schemes in Chapter 5 proved that network congestion on fog clus-

ters can be significantly lowered without degradation on multiobjective metrics. In this chapter, we also measure the impact of lowered network congestion on the ARTT and the WARSP metrics with extended algorithms including proposed LAMOMRank algorithm.



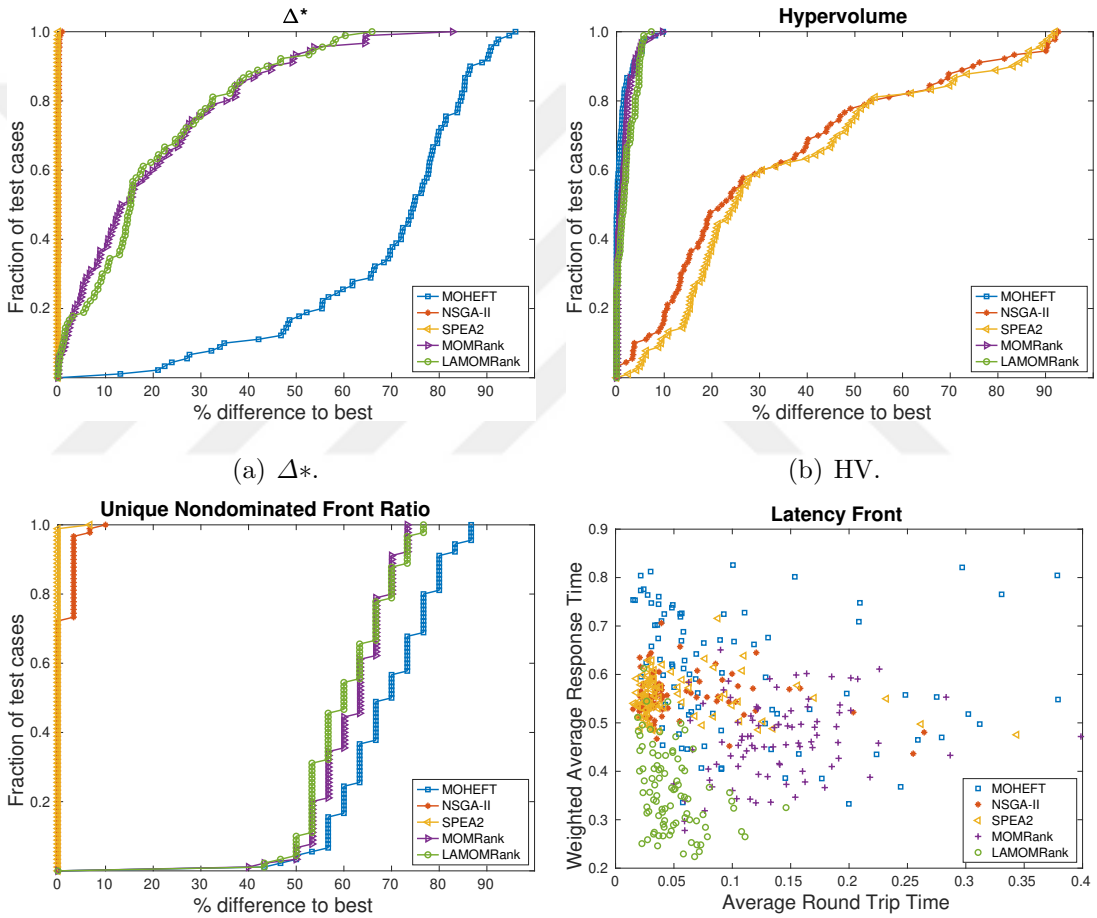
(a) $\Delta*$. (b) HV.

(c) UNFR. (d) Latency Front(WARSP vs ARTT).

Figure 6.5. Algorithm performances on cluster 1 for multi-objective metrics $\Delta*$, *HV*, *UNFR* and Latency Front for DeFog tasks(WARSP vs ARTT).

Figure 6.5 illustrates performance of the algorithms on cluster 1 for multiobjective metrics and presents latency front to emphasize and visually depict impact of LAMOMRank algorithm on ARTT and WARSP metrics. MOHEFT algorithm has the best performance for HV metric on cluster 1, while NSGA-II and SPEA2 do not converge enough with limited amount of time as in Chapter 5 yielding worst results (Figure 6.5(b)). The LAMOMRank algorithm has slightly better performance com-

pared to MOMRank algorithm on both HV and $\Delta*$ metric. MOHEFT, MOMRank and LAMOMRank perform similary for UNFR metric where NSGA-II and SPEA2 producing best values. The LAMOMRank algorithm has the best values in WARSP and ARTT metrics as depicted in Figure 6.5(d). We can clearly observe that latency front significantly pushed towards minimum values for latency metrics. The LAMOMRank algorithm serves to the motivation its proposed for, by lowering latencies without significant lose on multiobjective metrics.



(a) $\Delta*$. (b) HV.

(c) UNFR. (d) Latency Front(WARSP vs ARTT).

Figure 6.6. Algorithm performances on cluster 2 for multi-objective metrics $\Delta*$, *HV*, *UNFR* and Latency Front for DeFog tasks(WARSP vs ARTT).

Figure 6.6 illustrates results of the same experiment for multiobjective metrics and latency front on cluster 2. MOHEFT, MOMRank and LAMOMRank algorithms have almost identical performance on HV metrics (Figure 6.6(b)). MOMRank and LAM-

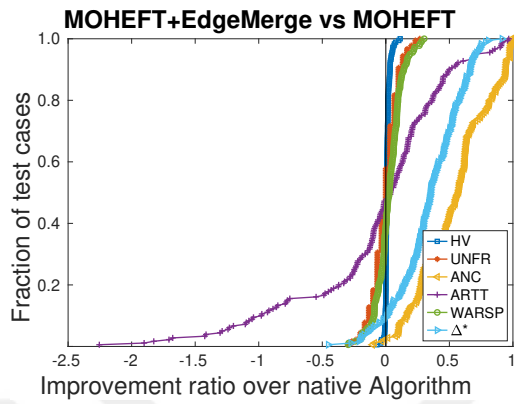
OMRank algorithms clearly outperform MOHEFT in UNFR metric (Figure 6.6(c)) in contrast to cluster 1 results (Figure 6.5(c)). LAMOMRank performs similar to MOMRank algorithm on cluster 2 for $\Delta*$ metric (Figure 6.6(a)) but we have a more clear performance clusters depicted on latency front (Figure 6.6(d)). LAMOMRank performs best on WARSP and ARTT metrics without introducing more complexity to MOMRank algorithm and performing similar on multiobjective metrics.

On another experiment, we measure the impact of task clustering schemes CompMatching and EdgeMerge on performance of the algorithms for all metrics. We observed on Chapter 5 that task clustering schemes can improve performance on various metrics even though being proposed for reducing average network congestion on fog clusters. We aim to measure performance change on altered test instances with latency constrained applications of DeFog benchmark for algorithms in Chapter 5 together with performance of the LAMOMRank algorithm which extends MOMRank algorithm.

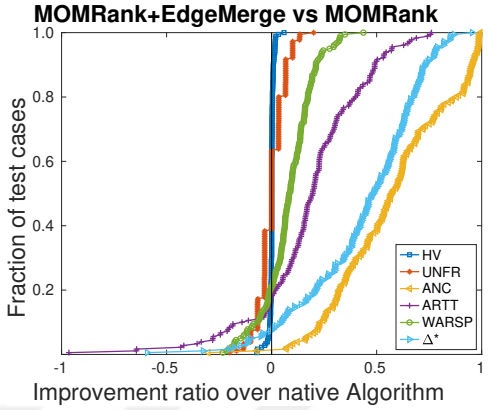
Figure 6.7 and 6.8 illustrates performance of native algorithms vs extended algorithms on all metrics including ARTT and WARSP over all instances as performance profiles for EdgeMerge and CompMatching extensions respectively. LAMOMRank algorithm performs consistently with MOMRank algorithm on multiobjective metrics HV,$\Delta*$ and UNFR for both extensions (Figure 6.7(b), 6.7(e), 6.8(b), 6.8(e)). However, we can observe that MOMRank benefits from task clustering extensions on both ARTT and WARSP. LAMOMRank algorithm seems to have almost no improvements on WARSP and little improvements on ARTT considering the accumulated area between y-axis for these metrics(Figure 6.7(e), 6.8(e)). NSGA-II algorithm has distinguished improvements on HV metric on CompMatching extension compared to EdgeMerge extension(Figure 6.8(c), 6.7(c)). LAMOMRank algorithm significantly improves ANC on both extension schemes performing similar to MOMRank algorithm.

When latency metrics (WARSP and ARTT) are considered, we observe no change on average performance for MOHEFT, NSGA-II and SPEA2 algorithms with both extensions in contrast to MOMRank and LAMOMRank. Our porposed algorithms

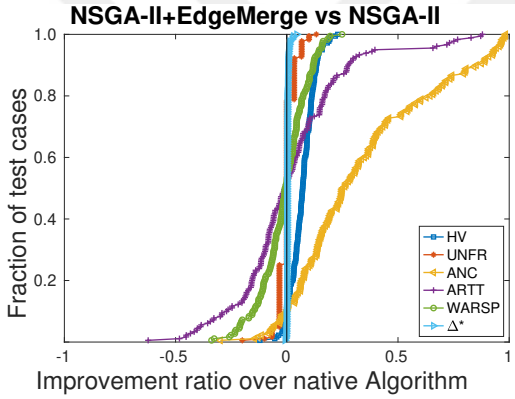
validate that addressing versatility of the fog scheduling problem with novel allocation decisions can provide improvements on many aspects of the problem.



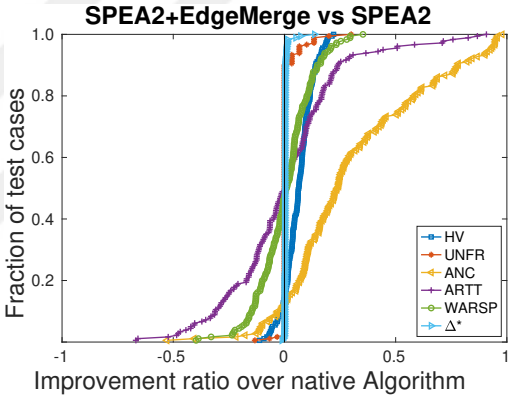
(a) MOHEFT.



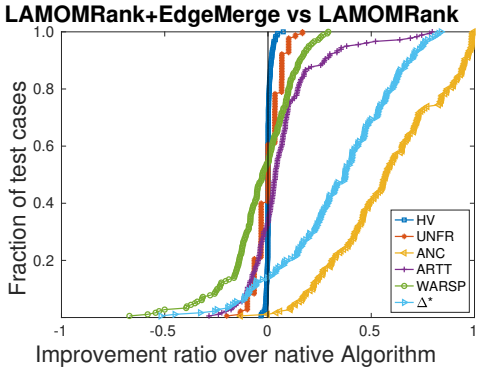
(b) MOMRank.



(c) NSGAII.

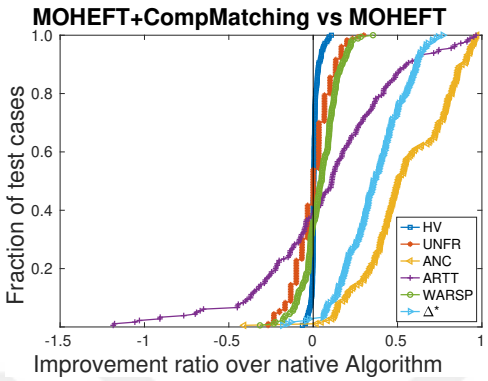


(d) SPEA2.

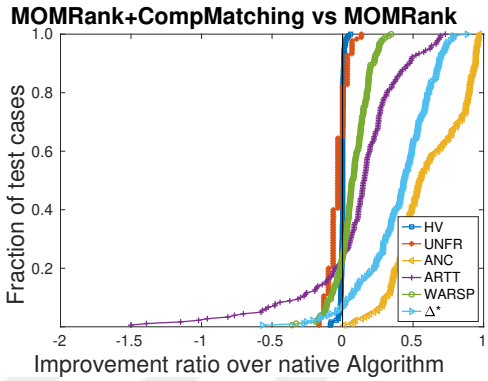


(e) LAMOMRank.

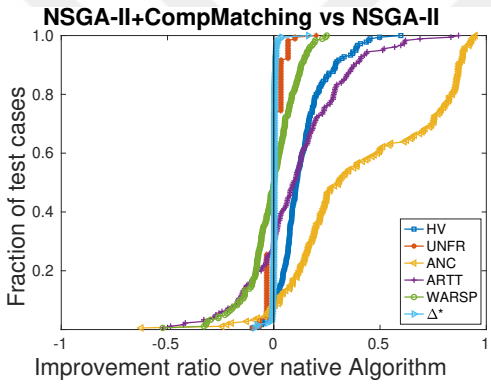
Figure 6.7. Impact of DAG clustering extensions for multi-objective metrics, ANC, WARSP and ARTT on algorithms with EdgeMerge extension.



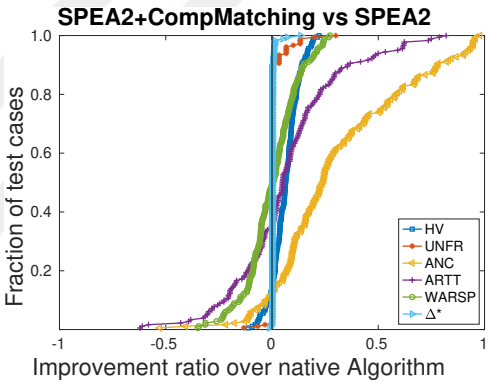
(a) MOHEFT.



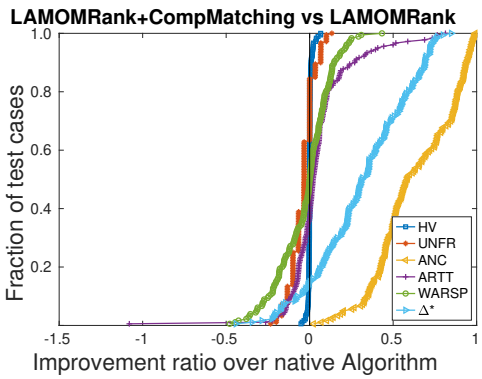
(b) MOMRank.



(c) NSGAII.



(d) SPEA2.



(e) LAMOMRank.

Figure 6.8. Impact of DAG clustering extensions for multi-objective metrics, ANC, WARSP and ARTT on algorithms with CompMatching extension.

7. CONCLUSIONS AND FUTURE WORK

In this thesis, we introduce two new multi-objective task scheduling algorithms for fog environments called the MOMRank and LAMOMRank algorithms, then compare their performance in terms of a set of multi-objective metrics and highlighted individual metrics with three prominent scheduling algorithms. Algorithms in our empirical evaluation is extended with two task clustering schemes, *EdgeMerge* which we propose and *CompMatching* from the literature, to minimize data movement of workflows on the network clusters aiming to reduce network congestion.

We conducted experiments on a well known workflow benchmark Pegasus first, then injected DeFog benchmark to emphasize real-time requirements in fog environments. Furthermore, we proposed extensions to the benchmarks with varying set of computation to communication ratio for further diversification in the problem instances. We used Amazon EC2 instances to create heterogeneous clusters for workflow computation. Our experimental study proves that added diversity with multiple task ranking to the scheduling algorithm(MOMRank) can improve for set of multi-objective metrics and network congestion can be further minimized. In addition, targeting response latency and round trip time as real-time application objectives (LAMOMRank) can improve latency front without degradation on other multiobjective fronts. Upcoming era of IoE introduces new challenges boosting complexity of the scheduling problems in heterogeneous fog environments. Our proposed algorithms prove that versatility of the problem can be tackled by diverting allocation decisions for the sake of many objectives or constraints without introducing more computational complexity.

As a future work, we are planning to dynamically change fog clusters due to possible failures in the environment. Another extension would be dynamically changing physical location of fog computing nodes to clusters where more computing sources are needed depending on dynamic service demand caused by user mobility.

REFERENCES

1. CISCO, "Cisco Annual Internet Report (2018–2023)", https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf, accessed on June 20, 2020.

2. Pedersen, M. V. and F. H. P. Fitzek, "Mobile Clouds: The New Content Distribution Platform", *Proceedings of the IEEE*, Vol. 100, No. Special Centennial Issue, pp. 1400–1403, 2012.

3. ETSI, "Mobile Edge Computing A key technology towards 5G", https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf, accessed on May 1, 2020.

4. Moustafa, H. and Y. Zhang, *Vehicular Networks: Techniques, Standards, and Applications*, Auerbach Publications, Boca Raton, FL, 2009.

5. Bonomi, F., R. Milito, J. Zhu and S. Addepalli, "Fog Computing and Its Role in the Internet of Things", *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, p. 13–16, Helsinki, Finland, 2012.

6. Puliafito, C., E. Mingozzi, F. Longo, A. Puliafito and O. Rana, "Fog Computing for the Internet of Things: A Survey", *ACM Transansactions on Internet Technology*, Vol. 52, pp. 71–99, 2019.

7. Ai, Y., M. Peng and K. Zhang, "Edge computing technologies for Internet of Things: a primer", *Digital Communications and Networks*, Vol. 4, No. 2, pp. 77–86, 2018.

8. Petri, I., A. R. Zamani, D. Balouek-Thomert, O. Rana, Y. Rezgui and M. Parashar, "Ensemble-based Network Edge Processing", *Proceedings of the 11th IEEE/ACM International Conference on Utility and Cloud Computing*, pp. 133–142, Zurich,

Switzerland, 2018.

9. Sarkar, S. and S. Misra, "Theoretical modelling of fog computing: a green computing paradigm to support IoT applications", *IET Networks*, Vol. 5, pp. 23–29, 2016.

10. Jalali, F., K. Hinton, R. Ayre, T. Alpcan and R. S. Tucker, "Fog Computing May Help to Save Energy in Cloud Computing", *IEEE Journal on Selected Areas in Communications*, Vol. 34, No. 5, pp. 1728–1739, 2016.

11. Chekired, D. A., L. Khoukhi and H. T. Mouftah, "Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory", *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 10, pp. 4590–4602, 2018.

12. Najafizadeh, A., A. Salajegheh, A. M. Rahmani and A. Sahafi, "Multi-objective Task Scheduling in cloud-fog computing using goal programming approach", *Cluster Computing*, Vol. 25, p. 141–165, 2022.

13. Huang, T., W. Lin, C. Xiong, R. Pan and J. Huang, "An Ant Colony Optimization-Based Multiobjective Service Replicas Placement Strategy for Fog Computing", *IEEE Transactions on Cybernetics*, Vol. 51, No. 11, pp. 5595–5608, 2021.

14. Wan, J., B. Chen, S. Wang, M. Xia, D. Li and C. Liu, "Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory", *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 10, pp. 4548–4556, 2018.

15. Singh, R. M., L. K. Awasthi and G. Sikka, "Towards Metaheuristic Scheduling Techniques in Cloud and Fog: An Extensive Taxonomic Review", *ACM Computing Surveys*, Vol. 55, No. 3, 2022.

16. Dastjerdi, A. V. and R. Buyya, "Fog Computing: Helping the Internet of Things Realize Its Potential", *Computer*, Vol. 49, No. 8, pp. 112–116, 2016.

17. Hu, P., H. Ning, T. Qiu, H. Song, Y. Wang and X. Yao, "Security and Privacy Preservation Scheme of Face Identification and Resolution Framework Using Fog Computing in Internet of Things", *IEEE Internet of Things Journal*, Vol. 4, No. 5, pp. 1143–1155, 2017.

18. Tang, B., Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He and Q. Yang, "Incorporating Intelligence in Fog Computing for Big Data Analysis in Smart Cities", *IEEE Transactions on Industrial Informatics*, Vol. 13, No. 5, pp. 2140–2150, 2017.

19. Basudan, S., X. Lin and K. Sankaranarayanan, "A Privacy-Preserving Vehicular Crowdsensing-Based Road Surface Condition Monitoring System Using Fog Computing", *IEEE Internet of Things Journal*, Vol. 4, No. 3, pp. 772–782, 2017.

20. Gupta, H., A. Vahid Dastjerdi, S. K. Ghosh and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments", *Software: Practice and Experience*, Vol. 47, No. 9, pp. 1275–1296, 2017.

21. Kraemer, F. A., A. E. Braten, N. Tamkittikhun and D. Palma, "Fog Computing in Healthcare–A Review and Discussion", *IEEE Access*, Vol. 5, pp. 9206–9222, 2017.

22. Ha, K., Z. Chen, W. Hu, W. Richter, P. Pillai and M. Satyanarayanan, "Towards Wearable Cognitive Assistance", *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, p. 68–81, Bretton Woods, New Hampshire, USA, 2014.

23. Zhu, J., D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu and F. Bonomi, "Improving Web Sites Performance Using Edge Servers in Fog Computing Architecture", *IEEE Seventh International Symposium on Service-Oriented System Engineering*, pp. 320–323, San Francisco, CA, USA, 2013.

24. Dave, G., X. Chao and K. Sriadibhatla, "Picaso Eigen Faces", `https://`

code.google.com/archive/p/picaso-eigenfaces/, accessed on June 20, 2020.

25. Hassan, M. A., M. Xiao, Q. Wei and S. Chen, "Help your mobile applications with fog computing", *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)*, pp. 1–6, Seattle, WA, USA, 2015.

26. Cao, Y., P. Hou, D. Brown, J. Wang and S. Chen, "Distributed Analytics and Edge Intelligence: Pervasive Health Monitoring at the Era of Fog Computing", *Proceedings of the 2015 Workshop on Mobile Big Data*, p. 43–48, Hangzhou, China, 2015.

27. Wang, X., Z. Ning and L. Wang, "Offloading in Internet of Vehicles: A Fog-Enabled Real-Time Traffic Management System", *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 10, pp. 4568–4578, 2018.

28. Sanchez, L., L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis and D. Pfisterer, "SmartSantander: IoT experimentation over a smart city testbed", *Computer Networks*, Vol. 61, pp. 217–238, 2014.

29. Zeng, D., L. Gu, S. Guo, Z. Cheng and S. Yu, "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System", *IEEE Transactions on Computers*, Vol. 65, No. 12, pp. 3702–3712, 2016.

30. Liu, Z., X. Yang, Y. Yang, K. Wang and G. Mao, "DATS: Dispersive Stable Task Scheduling in Heterogeneous Fog Networks", *IEEE Internet of Things Journal*, Vol. 6, No. 2, pp. 3423–3436, 2019.

31. Yang, Y., S. Zhao, W. Zhang, Y. Chen, X. Luo and J. Wang, "DEBTS: Delay Energy Balanced Task Scheduling in Homogeneous Fog Networks", *IEEE Internet*

of Things Journal, Vol. 5, No. 3, pp. 2094–2106, 2018.

32. Xiao, Z., X. Dai, H. Jiang, D. Wang, H. Chen, L. Yang and F. Zeng, "Vehicular Task Offloading via Heat-Aware MEC Cooperation Using Game-Theoretic Method", *IEEE Internet of Things Journal*, Vol. 7, No. 3, pp. 2038–2052, 2020.

33. He, Q., G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li and Y. Yang, "A Game-Theoretical Approach for User Allocation in Edge Computing Environment", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 31, No. 3, pp. 515–529, 2020.

34. Zeng, D., L. Gu and H. Yao, "Towards energy efficient service composition in green energy powered Cyber–Physical Fog Systems", *Future Generation Computer Systems*, Vol. 105, pp. 757–765, 2020.

35. Gu, L., W. Zhang, Z. Wang, D. Zeng and H. Jin, "Service Management and Energy Scheduling Toward Low-Carbon Edge Computing", *IEEE Transactions on Sustainable Computing*, Vol. 8, No. 1, pp. 109–119, 2022.

36. Sun, Y., F. Lin and H. Xu, "Multi-Objective Optimization of Resource Scheduling in Fog Computing Using an Improved NSGA-II", *Wireless Personal Communications*, Vol. 102, No. 2, p. 1369–1385, 2018.

37. Elaziz, M. A., L. Abualigah and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments", *Future Generation Computer Systems*, Vol. 124, pp. 142–154, 2021.

38. Téllez, N., M. Jimeno, A. Salazar and E. Nino-Ruiz, "A Tabu Search Method for Load Balancing in Fog Computing", *International Journal of Artificial Intelligence*, Vol. 16, pp. 106–135, 2018.

39. Ye, D., X. Wang and J. Hou, "Balanced multi-access edge computing offloading strategy in the Internet of things scenario", *Computer Communications*, Vol. 194,

pp. 399–410, 2022.

40. Zhang, L., K. Li, C. Li and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems", *Information Sciences*, Vol. 379, pp. 241–256, 2017.

41. Durillo, J. J., H. M. Fard and R. Prodan, "MOHEFT: A multi-objective list-based method for workflow scheduling", *4th IEEE Int. Conf. on Cloud Comp. Technology and Science Proceedings*, pp. 185–192, Taipei, Taiwan, 2012.

42. OpenFog, "OpenFog Consortium", `https://www.openfogconsortium.org/`, accessed on March 10, 2018.

43. OpenFog, "OpenFog Architecture", `https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf`, accessed on March 10, 2020.

44. McChesney, J., N. Wang, A. Tanwer, E. de Lara and B. Varghese, "DeFog: Fog Computing Benchmarks", *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, p. 47–58, Virginia, USA, 2019.

45. Sarkar, S., S. Chatterjee and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things", *IEEE Transactions on Cloud Computing*, Vol. 6, No. 1, pp. 46–59, 2018.

46. Mishra, S. K., D. Puthal, J. J. P. C. Rodrigues, B. Sahoo and E. Dutkiewicz, "Sustainable Service Allocation Using a Metaheuristic Technique in a Fog Server for Industrial Applications", *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 10, pp. 4497–4506, 2018.

47. Pham, X.-Q. and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system", *18th Asia-Pacific Network Operations and Management Symposium*, pp. 1–4, 2016.

48. Wu, H. and C. Lee, "Energy Efficient Scheduling for Heterogeneous Fog Computing Architectures", *IEEE 42nd Annual Computer Software and Applications Conference*, Vol. 01, pp. 555–560, Tokyo, Japan, 2018.

49. Niu, Y., Y. Liu, Y. Li, Z. Zhong, B. Ai and P. Hui, "Mobility-Aware Caching Scheduling for Fog Computing in mmWave Band", *IEEE Access*, Vol. 6, pp. 69358–69370, 2018.

50. Iosup, A., M. Jan, O. Sonmez and D. H. Epema, "On the dynamic resource availability in grids", *2007 8th IEEE/ACM International Conference on Grid Computing*, pp. 26–33, 2007.

51. Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197, 2002.

52. Zitzler, E., M. Laumanns and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization", pp. 95–102, Athens, Greece, 2001.

53. Topcuoglu, H., S. Hariri and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 3, pp. 260–274, 2002.

54. Herrmann, J., J. Kho, B. Uçar, K. Kaya and U. V. Catalyurek, "Acyclic Partitioning of Large Directed Acyclic Graphs", *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 371–380, Madrid, Spain, 2017.

55. Bharathi, S., A. Chervenak, E. Deelman, G. Mehta, M. Su and K. Vahi, "Characterization of scientific workflows", *2008 Third Workshop on Workflows in Support of Large-Scale Science*, pp. 1–10, Austin, TX, USA, 2008.

56. Jiang, S., Y. Ong, J. Zhang and L. Feng, "Consistencies and Contradictions of Performance Metrics in Multiobjective Optimization", *IEEE Transactions on Cybernetics*, Vol. 44, No. 12, pp. 2391–2404, 2014.

57. Fonseca, C. M., L. Paquete and M. Lopez-Ibanez, "An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator", *2006 IEEE International Conf. on Evolutionary Comp.*, pp. 1157–1163, Vancouver, BC, Canada, 2006.

58. Dolan, E. D. and J. J. Moré, "Benchmarking optimization software with performance profiles", *Mathematical Programming*, Vol. 91, pp. 201–213, 2002.

NOTLAR

Bu tez çalışması kapsamında ortaya çıkan ve telif hakkı yayınevine devredilen görseller, yayınevinin kendi ağ sayfasında bulunan "yazarın kendi ürettiği yazı ve grafiklerin tekrar kullanımı hakkında geçerli olan yayın politikası"na uygun olarak tez kitabında kullanılmıştır.