This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Latency-Aware Multi-Objective Fog Scheduling: Addressing Real-Time Constraints in Distributed Environments

LOKMAN ALTIN[1], HALUK R. TOPCUOGLU[2], (MEMBER, IEEE), AND FIKRET S. GÜRGEN.[3]

[1]Computer Engineering Department, Boğaziçi University, Istanbul, 34450 Turkey, (e-mail: lokman.altin@boun.edu.tr) and Siemens Advanta Turkey, Istanbul, 34870 Turkey, (e-mail: lokman.altin@siemens.com).
[2]Computer Engineering Department, Faculty of Engineering, Marmara University, Istanbul, 34854 Turkey.(e-mail: haluk@marmara.edu.tr).
[3]Computer Engineering Department, Boğaziçi University, Istanbul, 34450 Turkey, (e-mail: gurgen@boun.edu.tr).

ABSTRACT The fog computing paradigm was introduced to overcome challenges that cannot be addressed by conventional cloud computing, such as the lower response latency for real-time applications. Task scheduling in fog environments sets forth more complexity using novel objectives beyond scheduling in the cloud. In this study, a task scheduling model with five common objectives and two latency metrics is presented. We propose a latency aware multi-objective multi-rank scheduling algorithm, LAMOMRank, for fog computing. The performance of our algorithm was compared with that of three well known multi-objective scheduling algorithms, Non-dominated Sorting Genetic Algorithm (NSGA-II), Strength Pareto Evolutionary Algorithm (SPEA2) and Multi-objective Heterogeneous Earliest Finish Time (MOHEFT) algorithm, using three multi-objective metrics and two latency addressing metrics. We populate workload sets using Pegasus workflows and the DeFog benchmark to be distributed over two fog clusters generated with various Amazon Web Services instances. The empirical results validate the significance of our algorithm for better latency fronts including the response latency and task delivery time without performance degradation on multi-objective metrics.

INDEX TERMS Fog Computing, Task Scheduling, Latency-Constrained Applications, Multi-Objective Optimization, Multi-Objective Evolutionary Algorithms, Directed Acyclic Graphs

I. INTRODUCTION

Fog computing has emerged as a novel paradigm in response to evolving technology, particularly in addressing the critical demands posed by the proliferating Internet of Things (IoT), the rapid expansion of interconnected devices, and the advent of 5G connectivity. As the IoT ecosystem emerges with an anticipated connection of up to 1.2 trillion devices by 2030, the conventional cloud computing model has limitations, particularly in managing the enormous data volumes generated [1]. This paradigm shift towards fog computing has become vital for addressing the increasing demands for real-time processing, reduced latency, and enhanced security/privacy measures. Unlike traditional cloud setups, fog computing redistributes computational resources closer to the edge of the network by utilizing local devices such as routers, gateways, and localized cloudlets. The decentralized approach not only ensures efficient data processing but also introduces novel challenges, including uninterrupted service delivery for mobile devices, stringent security concerns, swift provisioning, reliability, cost-effectiveness, and energy efficiency, thereby emphasizing the critical requirements for fog computing in contemporary technological landscapes [2].

Task scheduling plays a significant role in optimizing the allocation of computational resources and addressing various challenges that emerge within fog computing environments. Fog scheduling constitutes strategic assignment and arrangement of tasks onto available fog computing nodes, considering various factors including computational load, data dependencies, resource constraints, and real-time requirements. Efficient task scheduling ensures optimal utilization of resources, enhances system performance, and addresses the dynamic nature of distributed architecture of fog computing. Various studies have attempted to emphasize the application latency for fog scheduling [3]–[5]. In most cases, there is a lack of a thorough definition of fog scheduling, which neglects a multitude of challenges. Fog scheduling models tend to concentrate on a singular objective or a limited set of weighted objectives defined in the unified metrics [6].

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

The low latency demand in real-world applications is one of the driving motivations for fog computing. Applications deployed to conventional cloud computing instances cause delays and low quality service for real-time requirements because they pass through many hop points for data transmission on a global network [2]. Latency-constrained applications must be allocated towards the edge of the network in the edge-to-cloud continuum to fulfill real-time, low latency demand.

In this study, we present a new *Latency-Aware Multi-objective Multi-Rank algorithm* (LAMOMRank) to prioritize latency-sensitive applications within a selectively real-time constrained workload set by extending the MOMRank algorithm. We propose a multi-objective fog scheduling model that attempts to address common objectives that are inherited from cloud scheduling, such as reliability, energy, makespan, and objectives novel to fog computing, including response latency and workload balancing. The LAMOMRank algorithm is then presented to address these challenges in a single scheduling framework.

The performance of the algorithm was assessed using two well-known multi-objective meta-heuristics: the Strength Pareto Evolutionary Algorithm (SPEA2) [7], Non-dominated Sorting Genetic Algorithm (NSGA-II) [8], and Multi-objective Heterogeneous Earliest Finish Time (MOHEFT) [9]. Furthermore, we extended the algorithms with two task clustering schemes to improve the latency front by reducing the data transfer on clusters. An experimental study was conducted on workflows generated from two benchmarks: the DeFog benchmark [10] and the Pegasus benchmark [11], where the former represents real-time applications and the latter represents interdependent tasks. The performance of the algorithms was measured using three multi-objective metrics unifying common metrics and three individual metrics on focus. The experimental results demonstrate the significance of our proposed LAMOMRank algorithm in improving the latency front without performance degradation on other multi-objective metrics.

The main contributions of this study are summarized as follows:

- We propose a Latency-Aware Multi-objective Multi-Rank (LAMOMRank) algorithm in heterogeneous fog computing environments for latency-constrained applications.
- An empirical study is designed for performance validation by generating various workloads with real-world applications and real-time-constrained applications in a workflow structure.
- In comparison to well-known multi-objective algorithms, the results of experiments demonstrate that our proposed LAMOMRank algorithm generates schedules with low latency without compromising other objective fronts.

The remainder of this paper is organized as follows. Section II summarizes related work. In Section III, we propose our multi-tier fog system model. Section IV details the multi-objective scheduling problem and our proposed LAMOMRank algorithm. In Section V, we discuss the experimental setup and workloads. Section VI provides the experimental results. Finally, the conclusions are presented in Section VII.

II. RELATED WORK

Fog computing extends cloud computing by distributing its functionality across diverse locations, presenting novel challenges that have yet to be addressed [12]. Its aim is to bring services towards the edge of the network via network devices such as routers and gateways to improve user experience. Hence, the advent of fog computing has brought forth a spectrum of emerging challenges encompassing security/privacy concerns, scalability, heterogeneity, reliability, and dynamic resource provisioning, while also inheriting traditional cloud computing challenges. Edge computing and IoT are also serving as complementary paradigms to fog computing, enhancing its capabilities by extending computational resources closer to the data source and facilitating the integration of a diverse range of interconnected devices.

IoT platforms play a significant role in enabling the integration and management of edge devices and indirectly complementing fog computing initiatives by facilitating the collection, processing, and analysis of data at the network edge. This integration of edge devices and services contributes to the overall efficiency and scalability of distributed computing environments. IoT platforms such as IBM IoT Cloud, AWS IoT Greengrass, and Microsoft Azure IoT Suite primarily provide comprehensive infrastructures and services for managing IoT devices and data [13]. IBM IoT Cloud offers advanced analytics capabilities and emphasizes interoperability with various protocols and devices, making it suitable for enterprises seeking to derive actionable insights from IoT data. AWS IoT Greengrass extends cloud capabilities to edge devices, enabling local data processing and management for improved latency and offline operation. Similarly, Microsoft Azure IoT Suite provides a wide range of services for building and deploying IoT applications, including device management, data analytics, and AI integration. The lack of widespread fog service providers offering pay-as-you-go infrastructure models, prompts to establish smaller-scale Edge/Fog computing environments using cloud service providers like Microsoft Azure and Amazon Web Services together with these IoT platforms [14]. Evaluating the real-world performance of IoT applications and various placement techniques in fog computing presents challenges distinct from Cloud computing, notably in managing distributed file systems with added effort and cost, particularly in large-scale scenarios.

Fog computing, with its decentralized approach, has emerged as a transformative paradigm that offers diverse applications across various domains. In the context of smart cities, fog computing enables real-time data processing at the edge, facilitating efficient traffic management, intelligent street lighting systems, and enhanced public safety through rapid video analytics [15]. IoT sensors used for data gather-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

ing, refining and exchanging with communication protocols at the near edge of the network are also vital in smart city implementations [16]. In the Industrial Internet of Things (IIoT), fog computing plays a pivotal role by enabling edge devices to process data locally, optimizing manufacturing processes, predictive maintenance in machinery, and ensuring seamless automation in industrial environments [17]. Furthermore, in healthcare, fog computing aids remote patient monitoring, facilitating timely data analysis and decision-making by healthcare professionals while ensuring data privacy and security [18]. From IoT perspective, particularly through the utilization of wearable sensors, it has significantly enhanced healthcare systems by enabling predictive and preventive healthcare measures to be implemented [19]. These scenarios illustrate how fog computing applications extend across domains, revolutionizing data processing and decision-making at the network edge.

Task scheduling in fog computing has been studied extensively in the literature, as taxonomically investigated in [20]. In [21], a Resource-Aware Cost-Efficient Scheduler(RACE) was proposed to address the cost, bandwidth, and makespan as individual metrics. Fog/cloud resources are classified with application modules into priority lists and allocated accordingly.The Dispersive Stable Task Scheduling Algorithm (DATS) was proposed to enhance the user experience by reducing latency through task offloading in diverse fog environments [22]. Processing efficiency (PE) concept first introduced by combining computing and communication capacities of the resources, then the DATS algorithm proposes concept of comprises two primary components: (1) PE-based resource selection and synchronized task scheduling. The experimental results demonstrated a reduction in service delay across fog nodes by effectively balancing computation resources and relationship capabilities. Chekired et al. [23] examined prioritized task scheduling within a multi-tier fog architecture, specifically testing it in a smart factory case scenario. Tasks were transferred to the upper-tier fog nodes within the architecture to evaluate the performance of the multi-tier fog architecture compared to the flat fog model, based on task completion and waiting times.

Arisdakessian et al. [24] introduced FoG-Match, an intelligent IoT-Fog scheduling methodology that utilizes game theory for task allocation among fog nodes. FoG-Match considers various objectives, including energy consumption, delay, and resource availability to make informed decisions. FoG-Match exhibits adaptability to changing conditions and learning capabilities based on past experiences. Learning-automata based dynamic fault-tolerant task scheduling (DFTLA) was proposed for fog computing [25]. Multiple objectives including energy and response time, were investigated in isolation, where the approach employed a state transition model to dynamically adapt task scheduling in response to the present system state and resource availability. A container-based task scheduling algorithm was proposed for smart manufacturing in fog environments [26]. The task completion time and response time objectives were investigated separately in the proposed approach, which also attempts to maximize concurrent service-containerized applications.

Meta-heuristics have been widely utilized in the literature to tackle the intricacy and adaptability challenges associated with multi-objective scheduling in fog environments [3], [27]–[30]. Multi-Objective Simulated Annealing (MOSA) algorithm is proposed, which is designed for the secure allocation of tasks across cloud and fog nodes, considering deadline constraints [3]. The Goal Programming Approach (GPA) was evaluated to determine a solution that met multiple objectives. Their approach includes consideration of access levels and scheduling aligned with client-requested goals for the allocation of IoT tasks between fog and cloud nodes. In [27], three algorithms employing heuristic marine predators were proposed to address the task scheduling challenge within a fog-cloud environment. In addition to the conventional Marine Predators Algorithm (MPA), the authors presented two additional versions: a modified MPA (MMPA) and an enhanced MMPA. The MMPA enhances the exploitability of the MPA algorithm by utilizing the most recently updated positions instead of solely relying on the last best position. Comparative assessments against various meta-heuristic algorithms were conducted, focusing on metrics such as energy consumption, makespan, carbon dioxide emission rate and flow time.

An Improved Firework Algorithm (I-FA) was proposed for fog scheduling targeting makespan and workload balancing [31]. Prior to scheduling, the tasks were clustered according to their storage, workload, and bandwidth. The I-FA algorithm improved the load balancing and completion time in a comparative study, and the objectives were analyzed independently.

Reducing the response latency and improving the service delivery time are key motivational aspects that are driving conventional cloud computing into fog era. Low response latency is a demand for real-time applications that are deployed and run on a remote server while users receive real-time network packets to interact. Conversely, service delivery time is mostly referred as to makespan, delivery time, execution time etc. which corresponds to the total execution time of an application with/without delivery times. These aspects have also been addressed in recent studies on fog scheduling [4], [5].

A scheduling algorithm that considers the application delay requirement in a fog-cloud environment was introduced in [4]. Their approach views each application as a collection of modules communicating with one another to accomplish tasks; when modules are situated far apart, it leads to a delay in the task completion time. EEG tractor beam game (EEGTBG) [32] was used as a real-time application demanding low response latency, and the proposed algorithm effectively handles the scheduling of real-time applications within multi-tiered cloudlet infrastructures. In another study [5], response latency was emphasized in hierarchical fog schemes for real-time dependent applications from IFogSim [32]. The

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

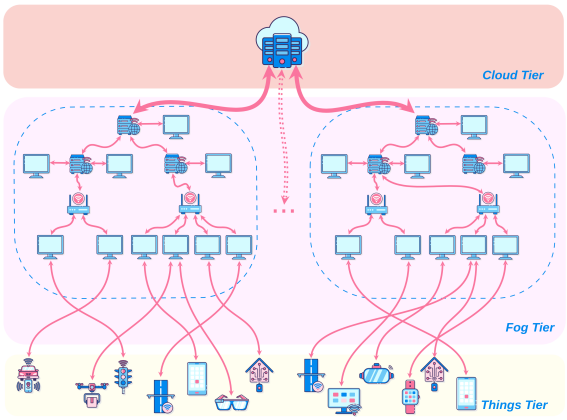


FIGURE 1: Architecture of Fog System Model

proposed algorithm aims to satisfy the latency requirements of the application by determining which service components should be relocated higher within the fog-cloud hierarchy. A communication-aware policy was introduced for resource allocation to prioritize access of resources among applications.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. SYSTEM MODEL

Our fog architecture is organized into three tiers: *Cloud Tier*, *Fog Tier* and *Things Tier* (see Figure 1). The *Fog Tier* comprises a structured arrangement of clusters, each depicted as a tree with varying depth levels. Tasks within a workflow can be distributed among edge nodes, fog nodes, or cloud nodes. It is essential to emphasize that workloads cannot be assigned to adjacent edge nodes because of privacy concerns, leading to reduced performance given the limited capacity of these edge nodes. Fog computing nodes are physically connected to local network devices such as gateways, switches, and routers. Our model operates under the assumption of abundant resources available in the *Cloud Tier*, with resource scarcity becoming more noticeable towards the network's edge. The *Things Tier* encompasses environments such as smart things, mobile vehicles, smart city devices, and wearable computing devices that are anticipated to access services provided by fog clusters.

B. PROBLEM FORMULATION

In this study, we outline five common objectives to be contained in a multi-objective model, four of which are minimization-based and one is maximization-based: minimizing makespan, energy consumption, costs, the degree of imbalance, and maximizing reliability within the system.

Workloads submitted to the system are depicted as directed acyclic graphs (DAGs), which illustrate the dependencies and transmission requirements among tasks. The model presented in this paper considers DAGs that consist of a single-exit task and a single-entry task. If a DAG contains multiple entry and/or exit nodes, we extend it by introducing pseudo-entry and/or exit tasks, each carrying zero workload and communication costs towards its predecessors and/or successors. The symbols used in this model are listed in the Table 1.

Symbol	Definition
w_m	Workload of task m
$FT(t_m)$	Finish time of task m
y_{mj}	Decision variable to allocate task m to node j
d_{mn}	Data dependency from task m to task n
T	Number of tasks
N	Number of computing nodes
L	Total number of physical link in a cluster
ρ_j	Processing capacity of node j
ϵ_j	Energy demand of computing node j per unit time
ζ	Total system energy consumption
R	Total system reliability
C	Total cost for computing
τ_{ij}	Bandwidth between nodes i and j

TABLE 1: Symbols used in the model

1) Makespan:

The initial objective aims to minimize the makespan, which quantifies the duration between submitting a workload and its completion. Equation 1 defines the makespan objective, utilizing $FT(t_{exit})$ to represent the finish time of the exit task t_{exit} within a specific Directed Acyclic Graph (DAG).

$$min.\ \ (FT(t_{exit})),\ \ where\ \ t_{exit} \in DAG \tag{1}$$

Every task must be assigned to either a cloud, fog, or edge computing node.

$$s.t.\ \ \sum_{j=1}^{N} y_{mj} = 1 \tag{2}$$

where m denotes the task number, and j represents the processing node ID, which can be a cloud, fog, or edge node. The allocation variable y_{mj} is set to 1 if task m is assigned to node j, and 0 otherwise. Our scheduling model operates under the assumption that task execution cannot be interrupted (non-preemptive).

$$FT(t_n) = \begin{cases} \left[max(FT(t_m) + \frac{d_{mn}}{\tau_{ji}}, ready_j) + \frac{w_n}{\rho_j}\right] & \text{, if } y_{ni} = 1 \text{ and } y_{mj} = 1 \\ \text{where } m \in pred(n) & \\ \frac{w_n}{\rho_j} & \text{, if } y_{ni} = 1 \\ \text{where } pred(n) = \emptyset & \end{cases} \tag{3}$$

In Equation 3, we compute the finish time (FT) of task n as the greater value between the machine's readiness time ($ready_j$) and the arrival of data dependencies from predecessor tasks. The workload of task n is denoted by w_n, whereas d_{mn} signifies the data transmission demand between task m and task n, where task m acts as a predecessor of task n (with $m \in$ pred(n)). The variables y_{ni} and y_{mj} represent the allocation of tasks (m, n) to the processing nodes across the cloud, fog, and edge tiers (i, j). The processing capacity of

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

node j is expressed as ρ_j, and the transmission capacity of the link connecting nodes i and j within each tier is defined using term τ_{ij}. The term $ready_j$ denotes the readiness time of node j, which is consistently assumed to be zero for cloud allocations, indicating the abundant availability of processing nodes.

2) Energy:

The second objective was to minimize energy consumption (Equation 4). The processing of tasks on computing nodes and transmission of data between these nodes constitute energy-intensive operations within the system.

$$min\ \zeta = \zeta^{process} + \zeta^{trans} \tag{4}$$

Equation 5 provides the total energy consumed during the processing of the computing nodes. Here, y_{mj} is the allocation decision (Equation 2), and ϵ_j stands for the energy consumed per unit of time on node j. The term w_m/ρ_j denotes the processing time of task m at node j.

$$\zeta^{process} = \sum_{m=1}^{T}\sum_{j=1}^{N} y_{mj} \times \epsilon_j \times \frac{w_m}{\rho_j} \tag{5}$$

Equation 6 aggregates the energy consumed during data transmission for all dependencies between tasks:

$$\begin{aligned} \zeta^{trans} = \sum_{m=1}^{T}\sum_{\substack{n=1\\ n\neq m}}^{T}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} \frac{d_{mn}}{\tau_{ij}} \times \epsilon_{ij}^{trans}, \\ s.t.\ y_{mi}=1, y_{nj}=1 \\ m \in pred(n) \end{aligned} \tag{6}$$

where d_{mn} stands for the data interdependence between task m and task n, while ϵ_{ij}^{trans} represents the energy utilized per time unit during data transmission from node i to node j. Notably, this model disregards the energy consumption during idle time of the processing nodes.

3) Cost:

The third objective focuses on minimizing costs, where Equation 7 provides the total expenditure incurred for allocation across the cloud, fog, and edge tiers. In this equation, c_j represents the cost per unit time of processing node j, and y_{mj} denotes an allocation decision.

$$min\ \ C = \sum_{m=1}^{T}\sum_{j=1}^{N} y_{mj} \times c_j \times \frac{w_m}{\rho_j} \tag{7}$$

4) Reliability:

In our model, the evaluation of reliability concerning both processing and data transmission relies on predetermined failure probabilities associated with every processing node and node-to-node transmission link. Our aim is to maximize the system reliability, calculated as the product of the total reliability in processing and the total reliability in data transmission.

$$max\ \ R = \prod_{m\in T} R^{process}(t_m) \times \prod_{d\in D} R^{trans}(d_{mn}) \tag{8}$$

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

We selected the Weibull distribution based on its close approximation in modeling resource failures, as referenced in [33]. The failure probability for each machine is specified in Equation 9, where η and β represent the scale and shape parameters of the Weibull distribution, respectively.

$$p(x) = \frac{\beta x^{\beta-1}}{\eta^{\beta}} e^{-(\frac{x}{\eta})^{\beta}} \tag{9}$$

We derive the failure rate for each task by integrating the failure probability associated with the processing node with respect to the time allocated for processing and transmission (Equation 10). The expression $\frac{w_m}{\rho_j}$ represents the duration of the time dedicated to executing task m on node j.

$$\begin{aligned} R^{process}(t_m) = 1 - \int_0^{\frac{w_m}{\rho_j}} p_j(x)dx = e^{(-\frac{\frac{w_m}{\rho_j}}{\eta})^{\beta}}, \\ s.t.\ y_{mj}=1 \end{aligned} \tag{10}$$

Equation 11 represents the reliability attributed to inter-task data transmission, where the expression p_{ij} denotes the failure rate of the link from fog node i to fog node j at a given time instant x.

$$\begin{aligned} R^{trans}(d_{mn}) = 1 - \int_0^{\frac{d_{mn}}{\tau_{ij}}} p_{ij}(x)dx = e^{-(\frac{\frac{d_{mn}}{\tau_{ij}}}{\eta})^{\beta}}, \\ s.t.\ y_{mi}=1,\ y_{nj}=1\ , \\ m \in pred(n) \end{aligned} \tag{11}$$

5) Degree of Imbalance:

The transition from cloud computing to fog computing is driven primarily by the need to transmit vast volumes of data generated by connected devices directly to the cloud. The transmission of this substantial amount of data has contributed to the slowdown of the global network. This challenge also extends to the fog tier, where an insufficient balance in task allocation within the fog cluster can result in the overloading of fog nodes closer to the edge of the network. A balance in the workload distribution within the fog cluster is essential to avoid congestion at the edge, ultimately enhancing responsiveness to incoming job requests.

Equation 12 represents the degree of imbalance in fog node j.

$$DI_j = \sum_{m=1}^{T}\sum_{j=1}^{N} y_{mj} \times \frac{w_m}{\rho_j} \tag{12}$$

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

$$min\ \ DI = \frac{DI_{max} - DI_{min}}{DI_{avg}} \tag{13}$$

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

where DI_{max}, DI_{avg}, and DI_{min} represent, the maximum, average, and minimum degrees of imbalance across nodes in the fog cluster, respectively.

While our proposed fog scheduling model addresses key objectives essential for efficient resource allocation and task scheduling in distributed fog computing environments, it is important to acknowledge several limitations and potential challenges. One aspect that we neglect is the dynamic provisioning of fog services, which involves dynamically scaling resources based on fluctuating workload demands. Additionally, the dynamism of workload submission poses a challenge, as the arrival pattern and characteristics of incoming tasks may vary unpredictably, necessitating adaptive scheduling strategies.

Moreover, security and privacy concerns remain critical considerations, particularly in decentralized fog environments where sensitive data may be processed and transmitted across distributed nodes. Ensuring robust security mechanisms and privacy-preserving techniques is essential to safeguard against potential vulnerabilities and breaches. Furthermore, the preemptive execution of workloads introduces complexities in scheduling decisions, requiring efficient algorithms capable of making timely and informed task allocation choices to optimize system performance. Addressing these challenges will enhance the efficacy and reliability of fog computing systems in real-world deployments.

IV. A LATENCY AWARE MULTI-OBJECTIVE TASK SCHEDULING ALGORITHM

Multi-objective optimization (MOP) comprises the simultaneous optimization of multiple conflicting objectives, aiming to find solutions that offer trade-offs among these objectives rather than a single optimal outcome. For a MOP, the Pareto-optimal set (POS) is defined as a collection of solutions that are not dominated by any other solution in the decision space. Correspondingly, the Pareto-optimal front (POF) signifies the set of solutions determined within the objective space. Formally, a MOP is expressed as:

$$min \quad F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x})) \tag{14}$$

where $f_1(\vec{x})$ is the first objective, m is the number of objectives.

In this study, we propose the LAMOMRank algorithm, which feature prominent low-latency constraints for real-time applications. Two latency objectives were defined, which emphasizes application delivery and real-time response latency. Furthermore, the common objectives of energy, makespan, reliability, cost, and workload balance given in Section III are, unified as a multi-objective problem optimized synchronously. The following section describes the proposed LAMOMRank algorithm in detail.

A. LAMOMRANK ALGORITHM

We propose the LAMOMRank algorithm to address the varied perspectives of heterogeneous fog scheduling, considering the complex nature of scheduling in fog environments due to the abundance of objectives and the heterogeneity of resources. In the LAMOMRank algorithm, we divert the allocation decisions to favor real-time applications with the aim of minimal or no performance degradation on other objective fronts in the search space. Multiple task-ranking schemes are used to give precedence to task allocations; and real-time application allocations are limited to direct paths from submitter edge nodes to the cloud. Parallel sets of sub-schedules constructed iteratively in isolation and crowding distance based selection mechanisms were used.

Algorithm 1: LAMOMRank algorithm.

Input: W=(A,D), A=$\bigcup_{i=1}^{n} A_i$, Workflow application,
R=$\bigcup_{i=1}^{m} R_i$, Set of resources,
K, Number of tradeoff solutions,
P, Number of ranks.
Output: Non-dominated solution set S

```
1  function LAMOMRank(W, R, K)
2      forall p ∈ P do
3          Rank ← RANK_p(W)
4          forall k ∈ K/P do
5              S_pk ← ∅
6              for i ← 1, n do
7                  S'_p ← ∅
8                  for j ← 1, m do
9                      for k ← 1, K/P do
10                         if LATENCY_CONSTRAINED(A_i) then
11                             if ON_PATH_TO_CLOUD(R_j) then
12                                 S'_p ← S'_p ∪ {S_pk ∪ (Rank_i, R_j)}
13                             else
14                                 S'_p ← S'_p ∪ {S_pk ∪ (Rank_i, R_submitter(A_i))}
15                         else
16                             S'_p ← S'_p ∪ {S_pk ∪ (Rank_i, R_j)}
17                 S'_p ← SORT_CROWD_DIST(S'_p, K/P)
18                 S'_p ← FIRST(S'_p, K/P)
19     S = ⋃_{p=1}^{P} S_p return S
```

The Algorithm 1 presents the steps of the LAMOMRank algorithm. It iterates over all ranking schemes that assign precedence to all tasks (e.g. Pegasus and DeFog) in Line 2. For each ranking scheme, partial schedules are constructed by first iterating over tasks (Line 6) and, then iterating over resources (Line 8). P represents the number of ranking schemes, and K is the total number of solutions to be delivered. Hence, K/P represents the number of tradeoff solutions for each ranking scheme partitioned and optimized in parallel. We divert the allocation decision of the DeFog applications, which are assumed to be real-time constrained, compared with the Pegasus tasks in Line 10.

If resource node R_j is not on the path from the submitter edge node to the cloud, the LAMOMRank algorithm does not allow allocation (Lines 11-14). A duplicate partial schedule

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

is added to the solution set S'_p with submitter edge node allocation in line 14.

In summary, the LAMOMRank algorithm aims to explore the latency front where two conflicting latency objectives considered, measured by Weighted Average Response Time (WARSP) vs Average Round Trip Time (ARTT) metrics which are explained in detail at Section V-C. The core strategy is to force real time applications, in our experimental case DeFog tasks, to be allocated only on the exact path from the submitter edge node to the cloud. Intuitively, it is expected that allocations diverted from this path will increase response latency (WARSP), while not much gain in the computation power (ARTT) for DeFog applications. Thus, the LAMOMRank strategy aims to eliminate redundant hops in the network.

The LAMOMRank algorithm evaluates $m \times K$ schedules in every iteration, resulting in a complexity of $O(P \times n \times m \times K/P)$, where P is total number of ranking schemes and K/P is size of solution set for each ranking scheme. Notably, the number of trade-off solutions (K) and task ranking schemes is negligibly lower than the number of tasks (m) and resources (n) respectively. As a result, the time complexity of the LAMOMRank algorithm remains $O(n \times m)$, as in both MOMRank [34] and MOHEFT [9].

B. CLUSTERING TASKS

In [34], two task-clustering schemes were proposed to extend multi-objective algorithms to eliminate costly data movement in the fog clusters: *EdgeMerge* and *CompMatching*. The main motivation for eliminating data movement in the clusters is to reduce network congestion which is strongly related to the delivery time of the corresponding tasks (e.g., the ARTT metric described in Section III). Hence, we employ these two task-clustering schemes to manage a range of simultaneous objectives within the problem.

In *CompMatching* [35], tasks are clustered based on their topological significance, creating coarser DAGs that avoid cyclicity. Our LAMOMRank algorithm and reference algorithms were extended with *CompMatching* cluster tasks until the number of vertices fell below a predefined threshold. The *EdgeMerge* strategy operates by sorting the edges based on their weights (indicating data dependency and potential data transmission). Subsequently, it selects the top E edges and allocates both source and destination tasks of these edges to the same physical node.

V. EXPERIMENTAL SETUP

The performance of the LAMOMRank algorithm is assessed against that of the MOHEFT [9], MOMRank [34], NSGA-II [8], and SPEA2 [7] algorithms in our empirical study. The MOHEFT algorithm is a multi-objective extension of the HEFT algorithm [36]. The MOMRank algorithm was recently proposed to address multi-objective heterogeneous scheduling in fog environments. Furthermore, two widely used multi-objective evolutionary algorithms from the literature, the NSGA-II and the SPEA2 algorithms, were also considered to validate the performance of our LAMOMRank algorithm. We assume that the MOHEFT algorithm is the baseline algorithm, and all other algorithms' execution times are limited to the MOHEFT's execution time.

The algorithms were implemented in the C programming language and executed on a single device. To ensure fair comparisons, identical operators and parameter values were employed for all algorithms, unless specified otherwise. Recombination was performed using a single-point crossover technique, while two mutation operations were utilized with equal probability: randomly assigning a new node to a task and repositioning a task within the allocation order. The mutation probability was set at 0.3, and the merged edge count E was established as 10. As explained in Section III, any task within the workflow has the flexibility to be assigned to a cloud node, fog node, or submitter edge node.

A. WORKLOADS AND FOG CLUSTERS

We evaluate the performance of the algorithms using the DeFog benchmark [10] and workflows from the Pegasus Workflow Management System [11]. Pegasus workflows comprise a diverse set of dependent tasks that can be easily partitioned and distributed over fog clusters. There were five different workflows: CyberShake, Montage, Inspiral, Epigenomics, and Sipht. Although the Pegasus benchmark has various properties that are easily scalable and deployable for distributed computing, it lacks real time features. The single task delivery time or host response latency is not an objective or constraint for a task in Pegasus. The DeFog benchmark is selected for extending Pegasus workflows as a bag-of-task paradigm to represent applications with real-time delivery and response constraints.

The DeFog benchmark provides a total of six applications for the Fog Computing paradigm and three deployment modes: *cloud-only*, *edge-only* and *cloud-edge (Fog)*. Additionally, it has the prominent feature of presenting the comprehensive set of metrics from both the application and platform perspectives. We select three applications from the DeFog benchmark to generate 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, and the system estimates the 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 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 performs automatic synchronization of text and audio segments. The process of creating synchronization maps for a collection of text excerpts and audio files

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

that containing the corresponding text is commonly known as forced alignment. In this context, Aeneas identified and established a mapping between the relevant audio segments for each provided text asset. DeFog applications are provided as docker containers that can be easily deployed and run on any edge, fog, or cloud node.

We consider DeFog and Pegasus executions collected on the Amazon EC2 m5.large instance. The upload image size in MB, execution time in seconds, and download results in MB metrics are collected for DeFog applications. These metrics are fed into the workload space together with the Pegasus runtimes to create a bag of tasks as illustrated in Figure 2. We form workloads to be deployed on our two fog clusters. Amazon Elastic Compute Cloud (Amazon EC2) instances are used in clusters that offer a diverse range of scalable capacities for CPU, memory storage, and networking within the cloud, catering to various application profiles. Details related to the computing capacity (ρ), network bandwidth (τ), cost (C), and reliability (β) of these instances are listed in Table 2. In most cases, units are negligible, and for multi-objective metric calculations (refer to Section V-B), each metric is normalized into the [0,1] range.

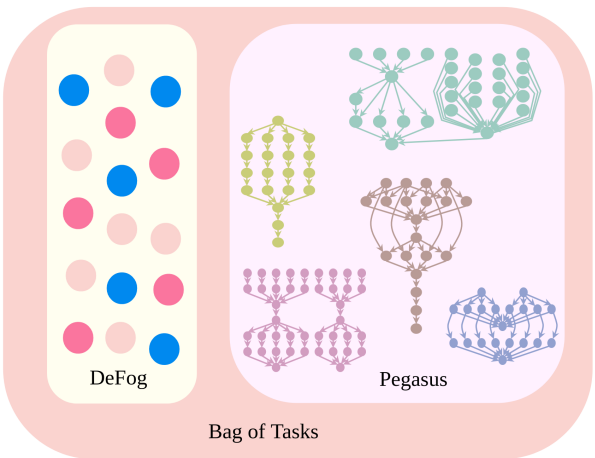


FIGURE 2: *Bag of Tasks* populated with Pegasus workflows and DeFog benchmark.

We scale DeFog applications workload and data (container image to be uploaded, results to be downloaded) to create a balance with Pegasus workflow instances, both in terms of total workload and number of tasks. We assume that workloads are submitted before scheduling, causing non-dependent parallel workloads (DeFog vs Pegasus) to populate each instance. A repair operation on problem instances is required, as in the case of the Pegasus workflows with multiple entry and/or multiple exit tasks. This is because the Bag of Tasks workload model enables parallel executions where there is no dependency between Pegasus and DeFog. Figure 3 depicts an example repair operation for system model consistency with the injection of pseudo entry/exit tasks and pseudo data dependencies between non-dependent tasks.

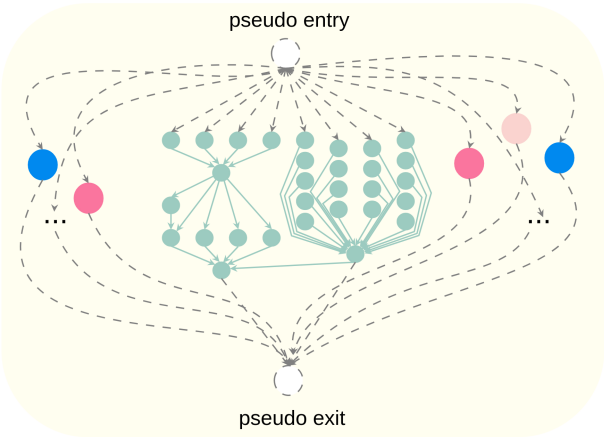


FIGURE 3: An example repair operation for workload model consistency.

We generate our test workflows using five applications of Pegasus for up to 100 tasks, where three workflows exist for each application. Then, we further modify these 15 workflows with varying computation to communication ratios sets,{0.2,0.5,1,5,10}, producing total of 90 workflows. An equal number of scaled DeFog tasks are added to workflows, which relaxes the dependencies in the workflow, as expected. The experimental results are collected on two clusters, yielding a total of 180 test instances. We employ three ranking systems to organize tasks for MOMRank and LAMOMRank: *upward ranking*, *workload ranking*, and *computation-to-communication ranking*. Upward ranking [9] prioritizes tasks based on their proximity to the exit node while considering their workload. In workload ranking, we initiate from the entry node and sequentially assign ranks to tasks with respect to the workload with the task precedence constraint. In the computation-to-communication ratio, tasks are arranged according to their workload ratio concerning input and output data demands with respect to precedence.

We quantify the computational capacity in terms of Computing Units (CU) for each instance, representing the number of independent computing units available on the respective instance. This enables us to gauge task execution, which is determined by the single-node execution time within the Pegasus workflows or the Defog benchmark. For data size units in workflow executions, the measurement is in bytes per second. Table 2 outlines the network bandwidths, presented in Gbps.

In our experimental setup, we simulated two physical fog networks in a tree topology using Amazon EC2 computing nodes. These network clusters are illustrated in Figure 4 and 5. On the cloud side, we employed *m5.24xlarge* instances, benefiting from their abundance. Conversely, edge nodes are equipped with *m5.large* instances, where only the submitter edge node serves as a computing alternative to cloud and fog

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

	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

TABLE 2: Capacity configurations for nodes from Amazon EC2 subset

in our model. In the *Fog Tiers*, a range of instances from Table 2 was utilized, spanning from *m5.xlarge* to *m5.16xlarge*.

Network response latency values were collected using the *traceroute* and *mtr* commands on the Linux development PC. These tools can be used to diagnose routers' information and statistics as the network packets travel to the given destination IP. We sent Internet Control Message Protocol (ICMP) packets to Amazon Web Services eu-north-1 (Stockholm, Europe) and collected the statistics of response latencies at every hop point. The response latency of some hop points was gradually selected for use in the experimental study. Figures 4 and 5 depict experimentally used clusters with computational instances and 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 the user experience for real-time applications. In our experimental study, if a DeFog application is deployed to a fog or cloud computing node, the total response latency will accumulate on the path from the submitter node to the target computing node.

B. METRICS FOR MULTI-OBJECTIVE OPTIMIZATION

To evaluate the performance of algorithms concerning multi-objective optimization problems, numerous metrics are available in the existing literature [37]. In our empirical study, we focus on three extensively employed metrics: the Unique Non-dominated Front Ratio (UNFR), Δ^*, and the Hypervolume (HV).

Given that the true Pareto front remains unknown for these experiments, we generated a set of all non-dominated solutions using the evaluated algorithms as the approximate Pareto front, denoted as P. The set of solutions obtained by the algorithms is referred to as S in the subsequent metric definitions.

The *Unique Non-dominated Front Ratio (UNFR)* metric indicates a unique portion of non-dominated solutions in the solution set [37].

$$UNFR(S) = \frac{|s \in S_{unf} | \nexists r \in R_{unf} : r \prec s|}{|R_{unf}|} \quad (15)$$

In this equation, R_{unf} denotes the reference unique non-dominated front. We opt for a unique set S as R_{unf} by

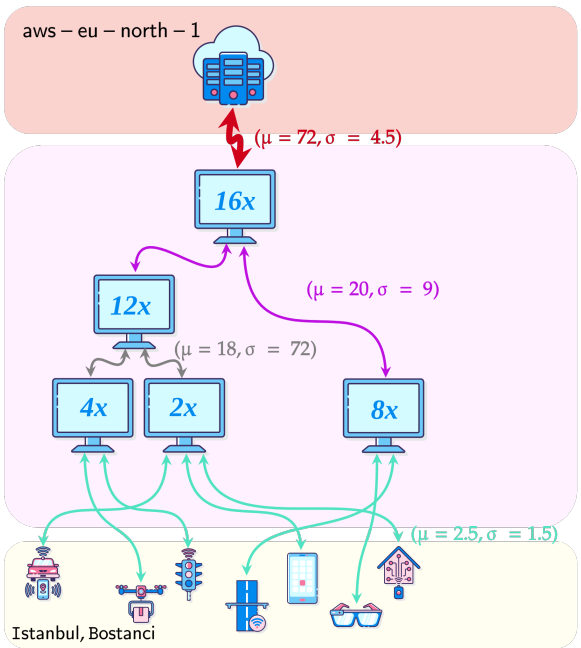


FIGURE 4: Cluster 1 with Amazon EC2 instances and latency measurements

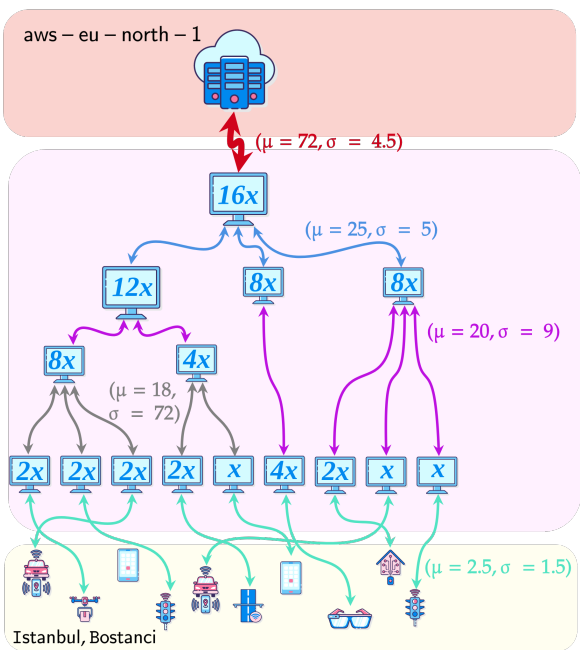


FIGURE 5: Cluster 2 with Amazon EC2 instances and latency measurements

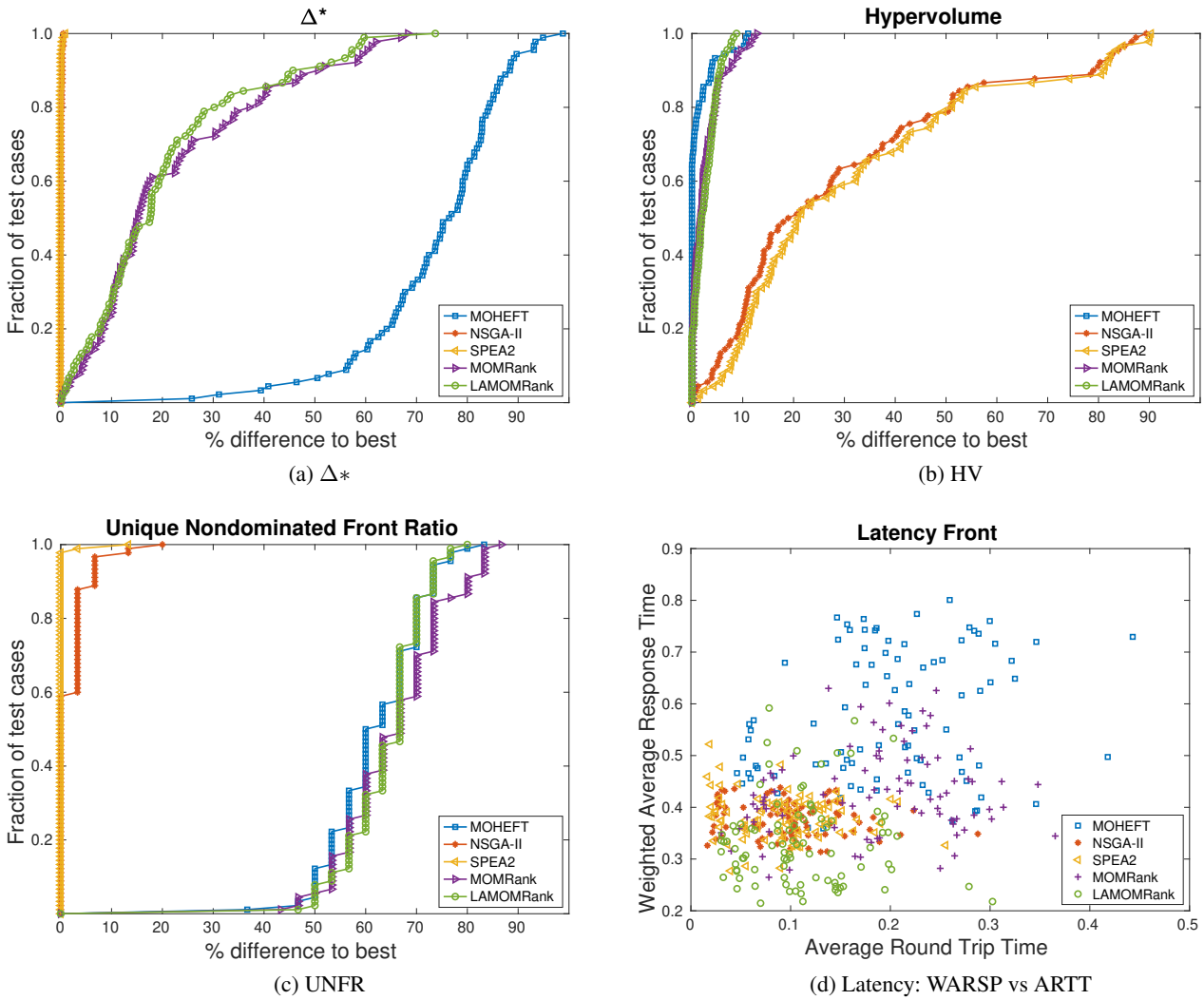
eliminating duplicate solutions; hence, the metric values fall within the range of [0,1].

Δ^* is another diversity metric which measures the spread value in the solution set S,

$$\Delta^*(S,P) = \frac{\sum_{k=1}^{m} d(\vec{e}_k, S) + \sum_{i=1}^{|S|} |d_i - \bar{d}|}{\sum_{k=1}^{m} d(\vec{e}_k, S) + (|S|)\bar{d}} \quad (16)$$

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664



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

(c) UNFR (d) Latency: WARSP vs ARTT

FIGURE 6: Algorithm performances on cluster 1 for multi-objective metrics $\Delta*$, *HV*, *UNFR* and Latency Front for DeFog tasks(WARSP vs ARTT).

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

The *Hypervolume (HV)* metric is one of the most frequently utilized multi-objective metrics in the literature, despite its considerable complexity [37]. The HV measures the volume within the search space, bounded by the known Pareto front. Owing to the disparate value ranges among different objectives, an unnormalized HV calculation may accentuate the impact of higher ranges. Therefore, we normalized each objective value using the known maximum and minimum values within the corresponding metric dimension.

C. METRICS FOR REAL-TIME APPLICATIONS

We assess the performance of the reference algorithms and the LAMOMRank algorithm with two new latency metrics for real-time constrained DeFog benchmark applications: *Weighted Average Response Latency* (WARSP) and *Average Round Trip Time* (ARTT).

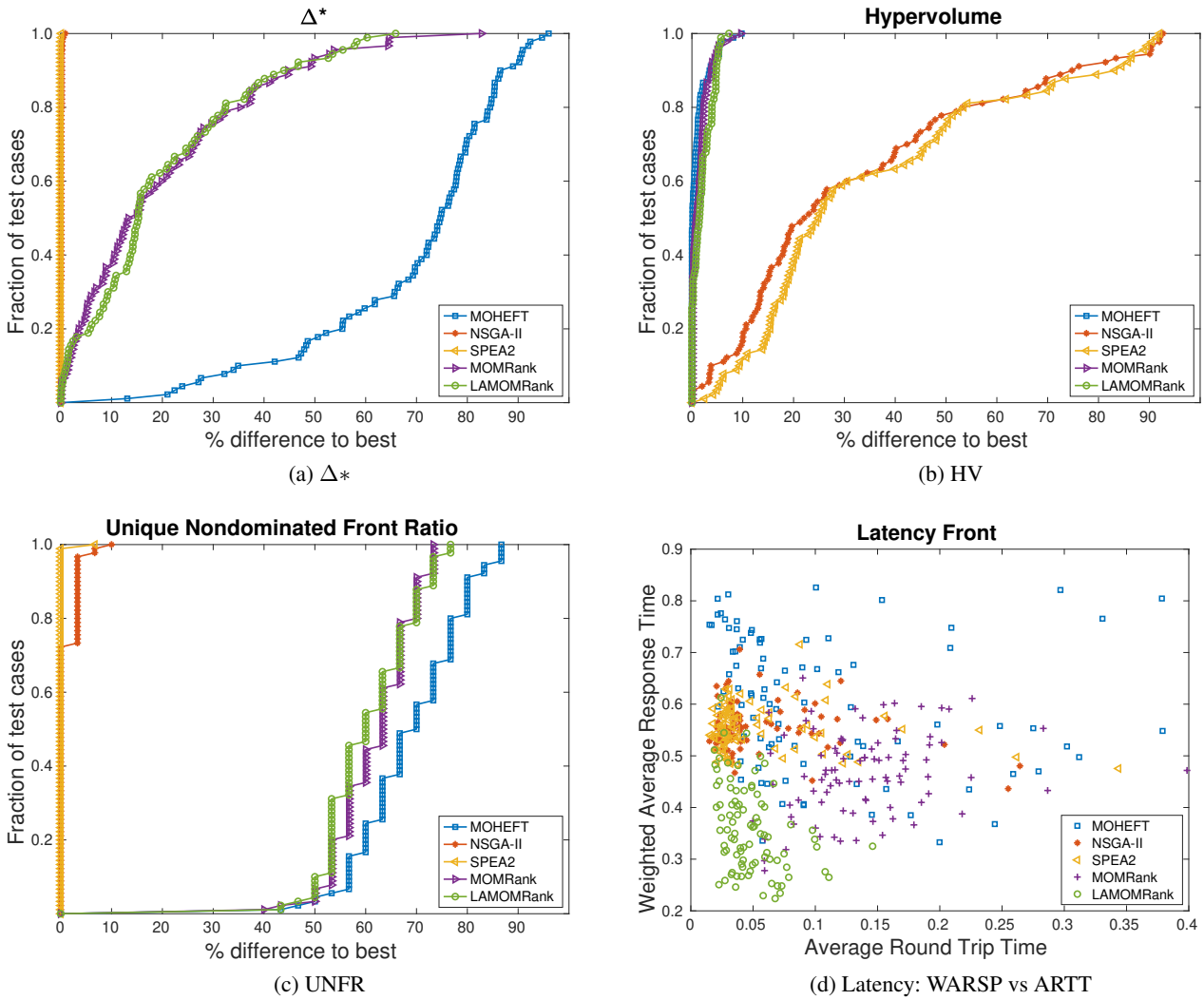
The WARSP metric can be a significant service requirement because live applications are deployed and run on a node where the user interacts with the application in real time. The *WARSP* metric is the average of the response latencies of DeFog applications weighted by the workload size. Response latencies used for each node in the experimental study for two clusters illustrated in Figure 4 and 5. The*WARSP* metric is calculated as:

$$WARSP = \sum_{m=1}^{T} \sum_{j=1}^{N} y_{mj} \times \Big(\sum_{l=1}^{L} \frac{response_latency_l}{w_m} \Big),$$
$$s.t. \;\; l \in Path(R_{submitter(m)}, R_j),$$
$$m \in \{DeFog\} \tag{17}$$

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 the submitter edge node ($R_{submitter}$) to

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664



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

(c) UNFR (d) Latency: WARSP vs ARTT

FIGURE 7: Algorithm performances on cluster 2 for multi-objective metrics $\Delta*$, *HV*, *UNFR* and Latency Front for DeFog tasks(WARSP vs ARTT).

allocated resource (R_j).

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.The *ARTT* metric was used to measure the delivery time of computational tasks offloaded to clouds or fog nodes. It measures the average time between the start of the task image upload to the computing node and the finish time of the output data download for latency constrained applications. It is given as:

$$\begin{aligned} RTT(m) &= FT(out_{\downarrow_m}) - ST(img_{\uparrow_m}), \\ ARTT &= \sum_{m=1}^{T} \frac{RTT(m)}{T}, \\ s.t.\ m &\in \{DeFog\} \end{aligned} \tag{18}$$

where $RTT()$ is the round trip time for a single task, $out_{\downarrow_m}$ is the finish time of the output data arrival, and $img_{\uparrow_m}$ is the starting time for the task image data upload. The ARTT is then calculated as the average over all latency-constrained tasks.

We also measure *Average Network Congestion* (ANC), which is naturally expected to be correlated with ARTT. The *ANC* metric on a fog cluster can be calculated as,

$$ANC = \frac{\sum_{l=1}^{L} busy_time(link_l)}{L} \tag{19}$$

where $busy_time(link_l)$ is the time spent for data transmission on the physical link $link_l$ in a cluster, and L is the number of physical links in a fog cluster.

VI. EXPERIMENTAL RESULTS

We present the experimental results using performance profiles [38] to compare the algorithm performance across all the problem instances. The performances are calculated against

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

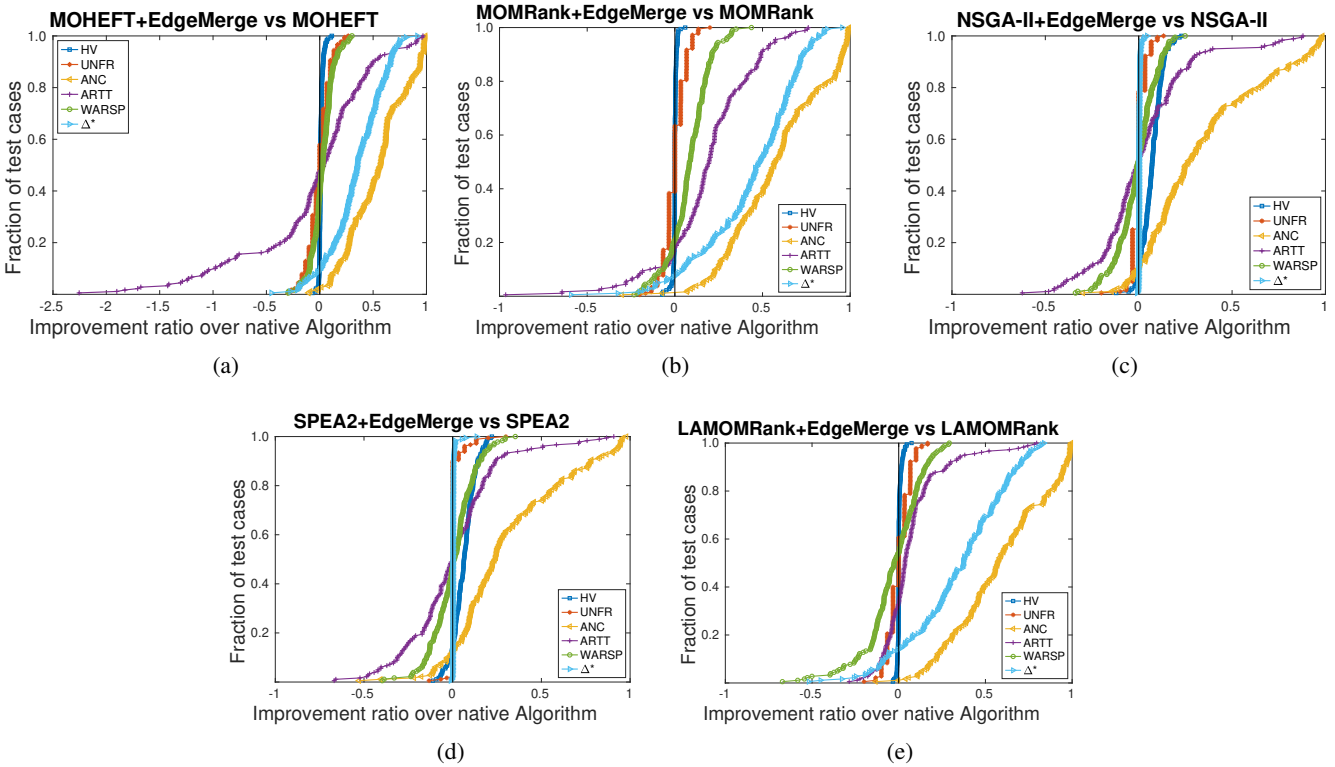


FIGURE 8: Impact of DAG clustering extensions for multi-objective metrics, ANC, WARSP and ARTT on algorithms with EdgeMerge extension.

the best result for the corresponding instance and plotted as sorted over all instances. Figure 6 illustrates the performance of the algorithms on cluster 1 for multi-objective metrics and presents the latency front to emphasize and visually depict the impact of the LAMOMRank algorithm on the ARTT and WARSP metrics. For the performance profiles, y-axis represents the fraction of test cases, whereas x-axis represents the difference to the best in percentage. Hence, the best-performing algorithm resides on y-axis, and the overall performance can be summarized as the minimization of the area between the y-axis and performance curves.

The MOHEFT algorithm has the best performance for HV metric in cluster 1, whereas NSGA-II and SPEA2 do not converge sufficiently with a limited amount of time, yielding the worst results (Figure 6b). The LAMOMRank algorithm performs slightly better than the MOMRank algorithm for both HV and $\Delta*$ metrics. MOHEFT, MOMRank, and LAMOMRank performed similarly for the UNFR metric where NSGA-II and SPEA2 produced the best values. The LAMOMRank algorithm had the best values in the WARSP and ARTT metrics, as depicted in Figure 6d. We can clearly observe that the latency front significantly pushed towards the minimum values for latency metrics. The LAMOMRank algorithm serves the motivation proposed for by lowering the latencies without significant degradation of multi-objective metrics.

Figure 7 illustrates the results of the same experiment for multi-objective metrics and the latency front of cluster 2. The MOHEFT, MOMRank, and LAMOMRank algorithms have almost identical performances in HV metrics (Figure 7b). The MOMRank and LAMOMRank algorithms clearly outperformed MOHEFT in the UNFR metric (Figure 7c) in contrast to the cluster 1 results (Figure 6c), which can be due to diverse ranking schemes becoming prominent in the expanded search space. LAMOMRank performs similarly to the MOMRank algorithm on cluster 2 for $\Delta*$ metric (Figure 7a) but we have clearer performance clusters on the latency front (Figure 7d). LAMOMRank performs best on WARSP and ARTT metrics without introducing more complexity to the MOMRank algorithm and performs similarly on multi-objective metrics.

In another experiment, we measured the impact of task clustering schemes CompMatching and EdgeMerge on the performance of the algorithms for all metrics, as proposed in [34]. The task clustering schemes improved performance on various metrics, such as reducing average network congestion on fog clusters, which is strictly related to the ARTT metric. In this experiment, each algorithm compared with its task clustering extension, naturally yielded worse results on some instances for some metrics. The overall performance improvement can be viewed as the total positive area between the performance curve and y-axis.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

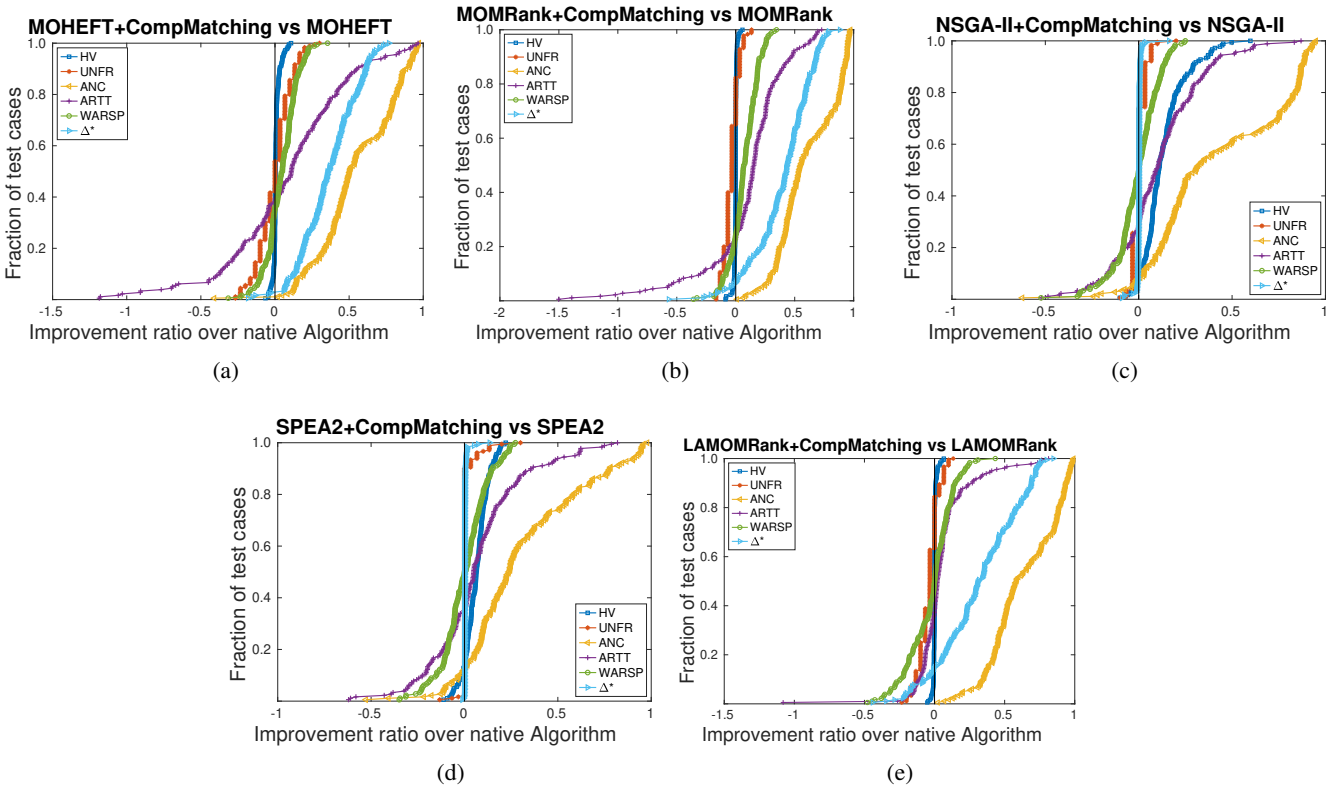


FIGURE 9: Impact of DAG clustering extensions for multi-objective metrics, ANC, WARSP and ARTT on algorithms with CompMatching extension.

Figures 8 and 9 illustrate the 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. Unlike the previous experiment, two cluster instances were accumulated in this experiment where 180 instances are plotted in each Figure. The motivation of task grouping extensions is to eliminate costly data movement on clusters by allocating large data-dependent tasks to the same physical node. This would relax the data movement on the cluster network; intuitively, one can expect a direct improvement on ARTT. The LAMOMRank algorithm performs consistently with the MOMRank algorithm on the multi-objective metrics HV, $\Delta*$ and UNFR for both extensions (Figure 8b, 8e, 9b, 9e). However, MOMRank benefits from task clustering extensions for both ARTT and WARSP. The LAMOMRank algorithm seems to have almost no improvement in WARSP, but significant improvement in ARTT considering the accumulated area between y-axis for these metrics(Figure 8e, 9e). The NSGA-II algorithm has distinguished improvements in the HV metric on the CompMatching extension compared with the EdgeMerge extension(Figure 9c, 8c). The LAMOMRank algorithm significantly improves the ANC on both extension schemes, performing similarly to the MOMRank algorithm. When latency metrics (WARSP and ARTT) are considered, we observe no change in the average performance for the MOHEFT, NSGA-II, and SPEA2 algorithms with both extensions, in contrast to MOMRank and LAMOMRank. Our proposed algorithm validate that addressing the versatility of the fog scheduling problem with novel allocation decisions can improve many aspects of the problem.

We categorize workflows by the number of tasks to emphasize the scalability of the LAMOMRank performance over three different instance sets. Small instance set contains workflows with around 60 tasks, medium instance set contains around 100 tasks and large instance set contains around 200 tasks. Each instance set is allocated on our two clusters, resulting total of six radar charts where complexity of the problem increases from Figure 10a to Figure 10f. We illustrate the result on radar chart using metrics HV, UNFR, $\Delta*$, ARTT and WARSP. Initially, the objectives HV, $\Delta*$, and UNFR are maximization metrics. Therefore, we convert them to errors with respect to the best-known values as in performance profiles. Consequently, all metrics are converted to minimization criteria. This experiment validates preeminence of NSGA-II and SPEA2 algorithms for generating most diverse non-dominated solution sets(UNFR) and best spread values ($\Delta*$). However, it is also clear that, these metaheuristics generating the worst HV values for each experiment set. LAMOMRank's performance is clearly distinguished when the problem instances become more complex with increasing number of tasks and the number of computation nodes on the cluster (from Figure 10a to Figure 10f). The area covered by LAMOMRank algorithm on these radar charts highlights the

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

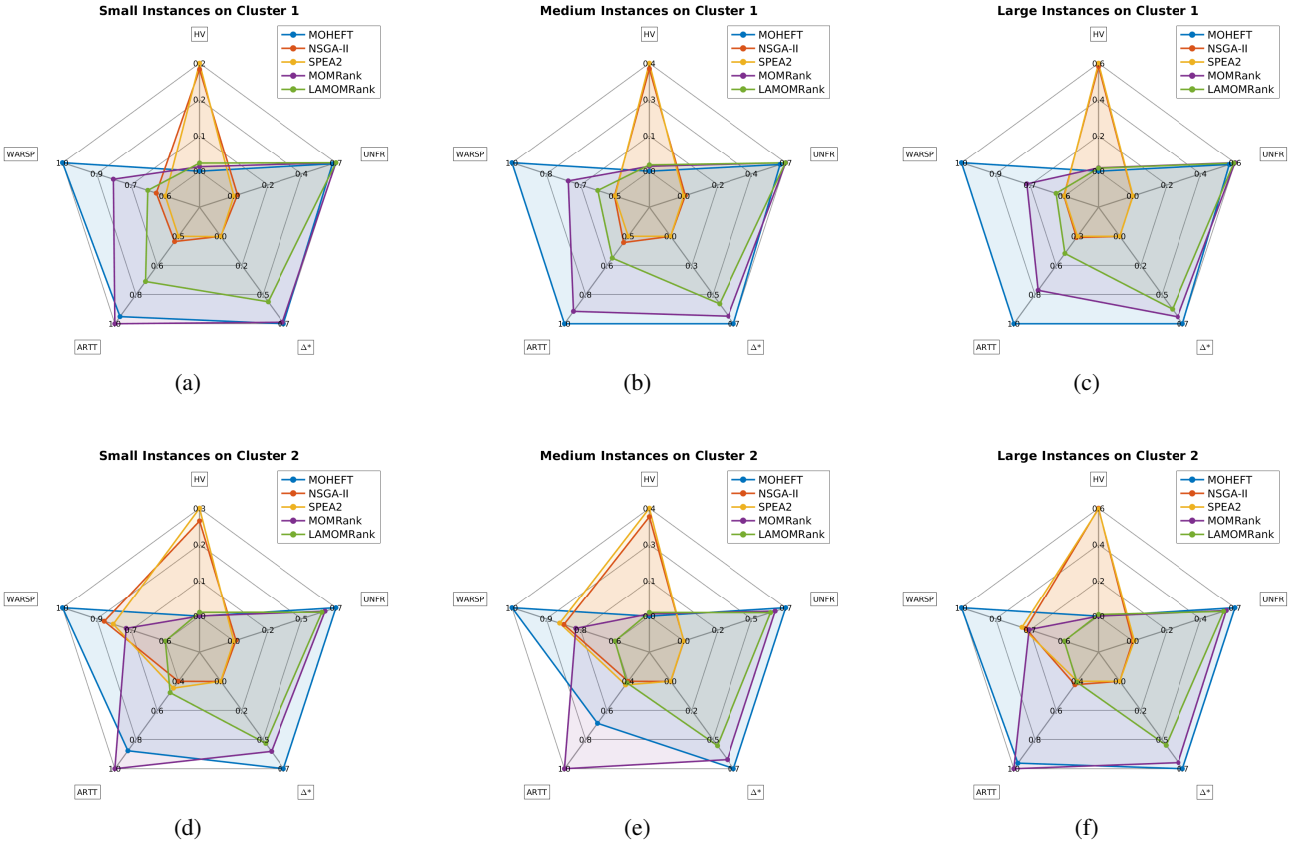


FIGURE 10: Performance comparison of algorithms on Cluster 1 and Cluster 2 for three instance sets with varying number of tasks

gap that our proposed algorithm targeted for scheduling in heterogeneous fog environments.

VII. CONCLUSION

Proposing a multi-objective task scheduling algorithm presents a significant challenge in the era of fog computing. Recent surveys [6], [20] indicate a scarcity of state-of-the-art algorithms capable of effectively addressing multiple objectives simultaneously. In this study, we proposed a fog scheduling model that attempt to address many of the open issues by considering makespan, energy, workload balance, reliability and cost as multi-objective metrics with the emphasize on two latency metrics, Weighted Average Response Time (WARSP) and Average Round Trip Time (ARTT). We selected workloads from the DeFog benchmark to represent real-time applications and the Pegasus benchmark as interdependent tasks that generate workflows. Our proposed LAMOMRank algorithm and task clustering schemes prove that without increasing the complexity of the algorithm, performance can be achieved on the latency front of multi-objective scheduling without degrading other objectives.

The effectiveness of our approach serves as evidence of the potential for innovative solutions to tackle the many challenges inherent in fog computing environments. By showcasing the efficacy of our algorithm in handling diverse objectives, we aim to inspire the development of new algorithms capable of addressing numerous open challenges simultaneously. This research not only contributes to advancing the field of fog computing but also underscores the importance of comprehensive approaches in addressing the complexities of modern computing paradigms. As a future work, we plan to extend our model with preemption to include mobility and enable intermittent services in fog environments.

REFERENCES

[1] CISCO, "Cisco annual internet report (2018–2023)," 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf
[2] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A survey," ACM Trans. Internet Technol., vol. 19, no. 2, apr 2019.
[3] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Multi-objective task scheduling in cloud-fog computing using goal programming approach," Cluster Comp., vol. 25, no. 1, pp. 141–165, Feb 2022.
[4] D. Charântola, A. C. Mestre, R. Zane, and L. F. Bittencourt, "Component-based scheduling for fog computing," in Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion, ser. UCC '19 Companion. New York, NY, USA: Association for Computing Machinery, 2019, p. 3–8. [Online]. Available: https://doi.org/10.1145/3368235.3368829
[5] M. L. M. Peixoto, T. A. L. Genez, and L. F. Bittencourt, "Hierarchical scheduling mechanisms in multi-level fog computing," IEEE Transactions on Services Computing, vol. 15, no. 5, pp. 2824–2837, 2022.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664

[6] R. M. Singh, L. K. Awasthi, and G. Sikka, "Towards metaheuristic scheduling techniques in cloud and fog: An extensive taxonomic review," ACM Comput. Surv., vol. 55, no. 3, feb 2022.

[7] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," vol. 3242, 2001.

[8] K. Deb, 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.

[9] J. J. Durillo, H. M. Fard, and R. Prodan, "Moheft: A multi-objective list-based method for workflow scheduling," in 4th IEEE Int. Conf. on Cloud Comp. Technology and Science Proceedings, 2012, pp. 185–192.

[10] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "Defog: Fog computing benchmarks," in Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, Virginia, USA, 2019, p. 47–58.

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

[12] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," IEEE Access, vol. 6, pp. 47 980–48 009, 2018.

[13] R. Sikarwar, P. Yadav, and A. Dubey, "A survey on iot enabled cloud platforms," in 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 2020, pp. 120–124.

[14] M. Goudarzi, M. Palaniswami, and R. Buyya, "Scheduling iot applications in edge and fog computing environments: A taxonomy and future directions," ACM Comput. Surv., vol. 55, no. 7, dec 2022. [Online]. Available: https://doi.org/10.1145/3544836

[15] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," ACM Comput. Surv., vol. 50, no. 3, jun 2017. [Online]. Available: https://doi.org/10.1145/3057266

[16] T. hoon Kim, C. Ramos, and S. Mohammed, "Smart city and iot," Future Generation Computer Systems, vol. 76, pp. 159–162, 2017.

[17] K. A. Abuhasel and M. A. Khan, "A secure industrial internet of things (iiot) framework for resource management in smart manufacturing," IEEE Access, vol. 8, pp. 117 354–117 364, 2020.

[18] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog computing in healthcare–a review and discussion," IEEE Access, vol. 5, pp. 9206–9222, 2017.

[19] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," IEEE Access, vol. 5, pp. 26 521–26 544, 2017.

[20] M. Hosseinzadeh, E. Azhir, J. Lansky, S. Mildeova, O. H. Ahmed, M. H. Malik, and F. Khan, "Task scheduling mechanisms for fog computing: A systematic survey," IEEE Access, vol. 11, pp. 50 994–51 017, 2023.

[21] J. U. Arshed and M. Ahmed, "Race: Resource aware cost-efficient scheduler for cloud fog environment," IEEE Access, vol. 9, pp. 65 688–65 701, 2021.

[22] Z. Liu, 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.

[23] D. A. Chekired, 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.

[24] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and N. Kara, "Fogmatch: An intelligent multi-criteria iot-fog scheduling approach using game theory," IEEE/ACM Transactions on Networking, vol. 28, no. 4, pp. 1779–1789, 2020.

[25] S. Ghanavati, J. Abawajy, and D. Izadi, "Automata-based dynamic fault tolerant task scheduling approach in fog computing," IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 1, pp. 488–499, 2022.

[26] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4712–4721, 2018.

[27] M. Abdel-Basset, D. El-Shahat, M. Elhoseny, and H. Song, "Energy-aware metaheuristic algorithm for industrial-internet-of-things task scheduling problems in fog computing applications," IEEE Internet of Things Journal, vol. 8, no. 16, pp. 12 638–12 649, 2021.

[28] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling," IEEE Access, vol. 7, pp. 116 218–116 226, 2019.

[29] M. Abdel-Basset, R. Mohamed, M. Elhoseny, A. K. Bashir, A. Jolfaei, and N. Kumar, "Energy-aware marine predators algorithm for task scheduling in iot-based fog computing applications," IEEE Transactions on Industrial Informatics, vol. 17, no. 7, pp. 5068–5076, 2021.

[30] H. Rafique, M. A. Shah, S. U. Islam, T. Maqsood, S. Khan, and C. Maple, "A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing," IEEE Access, vol. 7, pp. 115 760–115 773, 2019.

[31] S. Wang, T. Zhao, and S. Pang, "Task scheduling algorithm based on improved firework algorithm in fog computing," IEEE Access, vol. 8, pp. 32 385–32 394, 2020.

[32] H. Gupta, 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. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2509

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

[34] L. Altin, H. R. Topcuoglu, and F. S. Gürgen, "Network congestion aware multiobjective task scheduling in heterogeneous fog environments," IEEE Transactions on Industrial Informatics, pp. 1–10, 2023.

[35] J. Herrmann, J. Kho, B. Uçar, K. Kaya, and U. V. Catalyurek, "Acyclic partitioning of large directed acyclic graphs," in 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2017, pp. 371–380.

[36] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Transactions on Parallel and Distributed Systems, 2002.

[37] S. Jiang, 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.

[38] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," Mathematical Programming, vol. 91, no. 2, pp. 201–213, Jan 2002.



LOKMAN ALTIN received the B.S and M.S degrees in computer engineering from the Marmara University, Turkey, in 2012 and 2015. From 2013 to 2022, he was a Research Assistant in Marmara University Computer Engineering Department, Turkey. He is currently working as a Software Engineer at Siemens Advanta, Istanbul and pursuing a Ph.D. degree in computer engineering at Boğaziçi University, Turkey. His research interest includes task scheduling in fog computing, multi-objective optimization and evolutionary dynamic optimization.



HALUK RAHMI TOPCUOGLU is a Professor in Computer Engineering Department at Marmara University, Turkey. He obtained his Ph.D. degree from Syracuse University, Syracuse, NY in 1999. His research interests mainly include software-based hardware reliability, task scheduling and mapping for multicore architectures, scheduling in cloud computing and dynamic optimization problems. He is a member of the IEEE, the IEEE Computer Society and the ACM.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3395664



FIKRET SADIK GÜRGEN graduated from İstanbul Technical University with BS and MS degrees in Electronical Engineering Department in 1980 and 1982. He received MS degree from Ohio State University and Ph D degree from University of Akron in 1986 and in 1989.

He worked in NTT in Japan from 1989 to 1991. He joined the Computer Engineering Department of Boğaziçi University in 1991. He was a visiting Scientist in Sydney University Australia between 1993 and 1994. He was with Motorola Australian Research Center in 1995-1996. He was also a visiting Scientist in Macquarie university in Summer 1996.

He became a Professor of Computer Engineering Department in Boğaziçi University in 1998. He works in the same Department. He is the author of more than 100 publications.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4