# Network Congestion Aware Multiobjective Task Scheduling in Heterogeneous Fog Environments

Lokman Altin, Haluk Rahmi Topcuoglu, *Member, IEEE*, and Fikret Sadik Gürgen

***Abstract*—Task scheduling on fog environments surges new challenges compared to scheduling on conventional cloud computing. Various levels of heterogeneity and dynamism cause task scheduling problem is more challenging for fog computing. In this study, we present a multiobjective task scheduling model with a total of five objectives and propose a multiobjective multirank (MOMRank) scheduling algorithm for fog computing. The performance of the proposed strategy is assessed with well-known multiobjective metaheuristics [the nondominated sorting genetic algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm 2 (SPEA2)] and a widely used algorithm from the literature, the multiobjective heterogeneous earliest finish time (MOHEFT) algorithm using three common multiobjective metrics. Additionally, we incorporate two task clustering mechanisms to the algorithms in order to improve data transmissions on interconnection networks. Results of empirical evaluations given in performance profiles over all problem instances validate significance of both our algorithm and the integrated extensions for diminishing data transfer costs.**

***Index Terms*—Fog computing, metaheuristic techniques, multiobjective optimization, task scheduling, workflows.**

## NOMENCLATURE

| | |
|---|---|
| $w_m$ | Workload of task $m$. |
| $\mathrm{FT}(t_m)$ | Finish time of task $m$. |
| $y_{mj}$ | Allocation variable of task $m$ to node $j$. |
| $d_{mn}$ | Data movement requirement from task $m$ to task $n$. |
| $T$ | Number of tasks. |
| $N$ | Number of nodes. |
| $D$ | Number of data transmission on workflow. |
| $L$ | Number of physical link on a fog cluster. |
| $\rho_j$ | Processing power of node $j$. |
| $\epsilon_j$ | Energy consumption of node $j$ per unit time. |
| $\zeta$ | Total energy consumption. |
| $R$ | System reliability. |
| $C$ | Total cost. |
| $\tau_{ij}$ | Data transmission rate between nodes $i$ and $j$. |

Manuscript received 24 October 2022; revised 9 January 2023; accepted 22 July 2023. Date of publication 10 August 2023; date of current version 19 January 2024. This work was supported by the Boğaziçi University Research Fund (BAP) under Project BAP 21A01P3. Paper no. TII-22-4416. *(Corresponding Author: Haluk Rahmi Topcuoglu.)*

Lokman Altin is with Computer Engineering Department, Boğaziçi University, 34450 Istanbul, Turkey (e-mail: lokman.altin@boun.edu.tr).

Haluk Rahmi Topcuoglu is with Computer Engineering Department, Faculty of Engineering, Marmara University, 34854 Istanbul, Turkey (e-mail: haluk@marmara.edu.tr).

Fikret Sadik Gürgen is with Computer Engineering Department, Boğaziçi University, 34450 Istanbul, Turkey (e-mail: gurgen@boun.edu.tr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2023.3299624.

Digital Object Identifier 10.1109/TII.2023.3299624

## I. 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. 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 toward the edge of the network [2]. Network devices, such as routers and 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 [3].

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 Internet of Things (IoTs), smart factory comprising multiple tasks with dependencies [4], [5], [6], [7], most of them address single objective or a small set of weighted objectives. There are a few multiobjective 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.

In this study, we model the multiobjective 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 a novel multirank-based multiobjective scheduling algorithm (MOMRank) targeting versatility of the problem by utilizing existing multiobjective evolutionary algorithms and task scheduling algorithms. 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 to validate the effectiveness of the MOMRank algorithm, where it is compared with two well-known meta-heuristics [the nondominated sorting genetic algorithm II (NSGA-II) and the Strength Pareto Evolutionary Algorithm (SPEA2)] and a widely used method from the literature [the multiobjective heterogeneous earliest finish time (MOHEFT) algorithm]. The MOMRank algorithm significantly outperforms multiobjective metaheuristic algorithms and it performs comparable with the MOHEFT algorithm on hypervolume (HV) values. The MOMRank algorithm produces significantly better $\Delta*$ values compared to MOHEFT in less amount of time. To the best of authors' knowledge, this study is the first attempt to address five objectives and network congestion simultaneously as discussed and taxonomically depicted in a recent study [8].

The rest of this article is organized as follows. Section II summarizes the related work. In Section III, we propose our multitier fog scheme, the system model, and problem formulation. Section IV summarizes reference multiobjective algorithms, then we propose our novel approach in Section V. Section VI exhibits the experimental setup, and the results of empirical evaluations are given in Section VII. Finally, Section VIII concludes this article.

## II. Related Work

Fog computing is a decentralized computing model having many challenging aspects yet to be addressed [3]. Fog computing aims to bring computation and storage near edge of the network using network devices, such as gateways and 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 multitier architecture, mobility, and agile provisioning. Decentralized architecture of fog computing causes higher level of heterogeneity, which increases the complexity of the scheduling problem.

Fog models are proposed for various scenarios in the literature including smart manufacturing in industrial IoTs [4], [7], connected vehicle networks [9], and smart city [10]. 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 [11]. 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 multiobjective scheduling algorithms are proposed for fog computing in the literature [12], [13], they mostly emphasize only a subset of objectives covered in Section III either jointly or individually, lacking full coverage of all objectives given in this study. Dispersive stable task scheduling algorithm is proposed to increase user experience with latency reduction by offloading tasks in heterogeneous fog environments [12]. In their study, the two-phase algorithm is dedicated to first constructing coalition between fog nodes, helper nodes, and virtual machines; then it determines the task allocation vector of each task node. Yang et al. [13] proposed a novel delay energy balanced algorithm 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 the better delay performance.

Prioritized task scheduling in multitier fog architecture is tested for a smart factory case scenario by Chekired et al. [4]. Tasks are offloaded to the upper tier fog nodes in the architecture to signify performance of multitier fog architecture over flat fog model with task completion and waiting time. Xiao et al. [14] investigated vehicular task offloading and task allocation in mobile edge computing systems. The study targets utility maximization via cooperative game theoretic method using gathered real world traffic data. In [15], 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 [16], energy management in fog environments is formulated as a mixed integer linear programming, 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 is used in [17] 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 multiobjective domain to overcome the complexity of the problem [5], [6], [18], [19], [20]. Task latency and reliability are investigated using an improved version of the NSGA-II. algorithm [18]. Artificial ecosystem-based optimization enhanced with slap swarm algorithm is utilized for task scheduling of IoT applications in fog-cloud environments [19]. The performance of makespan and throughput objectives is evaluated individually, where improvements are depicted over other metaheuristics on real world and synthetic workloads. Multiobjective simulated annealing and goal programming are used together in [5] targeting service time, cost, deadline, and security objectives. A multiobjective ant colony optimization is proposed with multireplica service placement for scheduling in fog-IoT environments [6], where the deployment cost and the service latency are the objectives considered

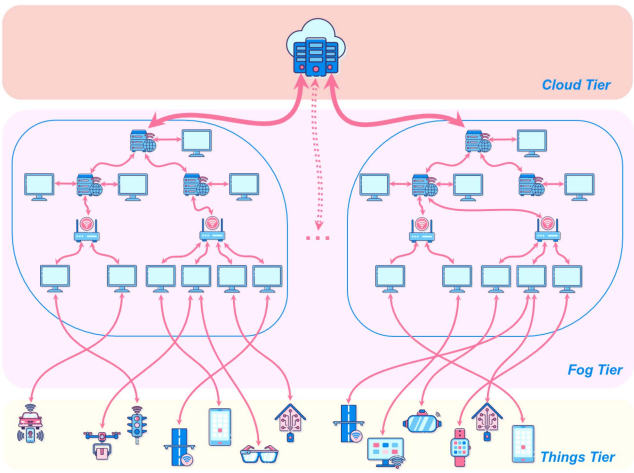


Fig. 1. Architecture of fog system model.

In another study, tabu search algorithm is used for load balancing between cloud and fog nodes in [20], which defines the problem as an integer linear programming problem. Ye et al. [21] proposed multiaccess edge computing offloading strategy by modifying the SPEA2 operators targeting both energy consumption and task execution delay objectives. Biobjective genetic algorithm is proposed to tackle energy consumption and system reliability for task scheduling in heterogeneous computing systems [22]. In their work, multiobjective differential evolution algorithm and MOHEFT algorithm are used as the reference algorithms. The MOHEFT [23], a list scheduling based algorithm, is one of the most commonly used multiobjective task scheduling algorithm in cloud-edge computing, which is also the baseline algorithm in our experimental study.

## III. System Model and Problem Formulation

### A. System Model

In our fog architecture, the system has the following three tiers: cloud tier, fog tier, and things tier (see Fig. 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 cannot be allocated on neighboring 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 toward edge of the network. In things tier; smart homes, mobile devices, smart city infrastructures, and wearable computing devices are expected to be receiving services from fog clusters.

### B. Problem Formulation

In this article, we formulate 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. The Nomenclature lists variables in our fog scheduling model.

*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. Equation (1) shows the objective of makespan where $\mathrm{FT}(t_{\mathrm{exit}})$ is the finish time (FT) of exit task $t_{\mathrm{exit}}$ of the given DAG

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

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

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

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 nonpreemptive in our scheduling model

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

In (3), we define FT of the task $n$, which is the maximum of machine ready time ($\mathrm{ready}_j$) 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 $n$ ($m \in \mathrm{pred}(n)$). 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 $\rho_j$ and the transmission capacity of link between nodes $i$ and $j$ in each tier is given as $\tau_{ij}$. The term $\mathrm{ready}_j$ represents the ready time of the node $j$, which is assumed always 0 for cloud allocations indicating processing nodes are available abundantly.

*2) Energy:* The second objective is the minimization of the energy consumption [see (4)]. Task processing of the computing nodes and data transmission between computing nodes are operations demanding energy in the system

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

Equation (5) gives total processing energy consumption of the computing nodes, where $y_{mj}$ represents allocation decision [see (2)], and $\epsilon_j$ is the energy consumption per time unit for node $j$.

The term $w_m/\rho_j$ represents the time spent for processing task $m$ on node $j$

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

Equation (6) accumulates transmission energy consumption for each data dependency between tasks

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

where $d_{mn}$ is the data dependency between task $m$ and task $n$; $\epsilon_{ij}^{\text{trans}}$ 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) Cost:* The third objective is the cost minimization, where (7) represents total cost spent for allocation to cloud, fog, and edge tiers. In this equation, $c_j$ represents cost per unit time of processing node $j$, where $y_{mj}$ is an allocation decision (2)

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

*4) Reliability:* In our model, we measure the reliability of processing and data transmission using predefined failure probability of each processing node and internode 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

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

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 [24]. Probability of failure for each machine is defined in (9), where $\eta$ and $\beta$ are scale and shape parameters of the Weibull distribution, respectively

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

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 (10). The term $\frac{w_m}{\rho_j}$ represents amount of time spent for executing task $m$ on node $j$

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

Equation (11) shows the reliability of intertask data transmission, where the term $p_{ij}$ represents failure rate of link from fog node $i$ to fog node $j$ at any time instant $x$

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

*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.

Equation (12) represents the degree of imbalance in fog node $j$

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

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

$$\min \quad \text{DI} = \frac{\text{DI}_{\text{max}} - \text{DI}_{\text{min}}}{\text{DI}_{\text{avg}}} \tag{13}$$

where $\text{DI}_{\text{max}}$, $\text{DI}_{\text{avg}}$, and $\text{DI}_{\text{min}}$ are the maximum, average, and minimum degree of imbalance in fog nodes, respectively.

In addition to these five objectives, we consider average network congestion (ANC) on fog clusters for fog network congestion extension of our MOMRank algorithm (see Section V-A). ANC on a fog cluster can be calculated as

$$\text{ANC} = \frac{\sum_{l=1}^{L} \text{busy\_time}(\text{link}_l)}{L} \tag{14}$$

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

## IV. Multiobjective Task Scheduling in Fog Computing

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

$$\min \quad F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) \tag{15}$$

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 [23], which is the baseline algorithm for

our experimental study. Two widely used multiobjective evolutionary algorithms from literature, the NSGA-II Algorithm [25] and the SPEA2 algorithm [26], are considered in the empirical evaluations, as well.

### A. MOHEFT Algorithm

MOHEFT [23] algorithm is a Pareto-based list scheduling algorithm extended from the classical task scheduling algorithm for heterogeneous resources, the heterogeneous earliest finish time (HEFT) Algorithm. 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 $m \times K$ 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.

## V. New Multiaspect Heuristic: MOMRank

In this article, we present a new task scheduling algorithm, called *multiobjective multirank (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 [23] 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 nondominated set of solutions among different objectives.

Algorithm 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 $m \times K$ 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 $O(n.m)$.

**Algorithm 1:** MOMRank.

**Input:** W = Workflow; R = Set of resources; K = Number of tradeoff solutions; P= Number of ranks.

1: **function** MOMRank(W,R,K)
2: **for** p ← 1, P **do**
3: Rank ← $RANK_p(W)$
4: **for** k ← 1, K/P **do**
5: $S_{pk} \leftarrow \emptyset$
6: **for** i ← 1, n **do**
7: $S'_p \leftarrow \emptyset$
8: **for** j ← 1, m **do**
9: **for** k ← 1, K/P **do**
10: $S'_p \leftarrow S'_p \cup \{S_{pk} \cup (Rank_i, R_j)\}$
11: $S'_p \leftarrow SORT\_CROWD\_DIST(S'_p, K/P)$
12: $S'_p \leftarrow FIRST(S'_p, K/P)$
13: S=$\bigcup_{p=1}^{P} S_p$
14: **return** S

### A. 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 [27]). In their work, a DAG clustering approach is proposed in the following 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 article, 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. The *CompMatching* algorithm marks edges iteratively, if topological order difference is less than 1 and the destination vertex does not have multiple sources, which would cause cyclicity.

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. Algorithm 2 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

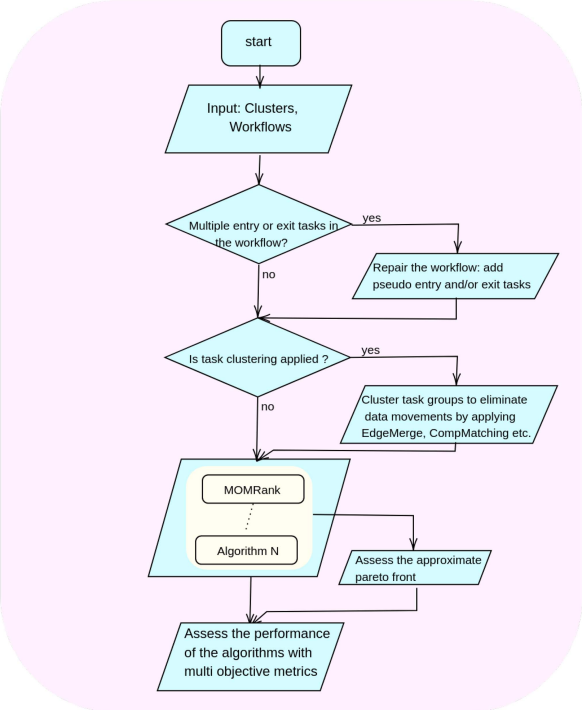


Fig. 2. Flow chart of our multiobjective scheduling framework for fog environments.

**Algorithm 2:** EdgeMerge.

**Input:** Workflow W= (T,D), tasks in T, edges as task dependencies in D, number of edges to be merged in E

```
function (W)
Cluster[] ← T   ▷ Initially each task in distinct cluster
D* ← Sort(D)
for e ← 1, E do
    src ← D*[e].source
    dst ← D*[e].destination
    Cluster[src]= Cluster[src] ∪ Cluster[dst]
return Cluster[]
```

allocate corresponding bundled tasks to the same new physical node.

Fig. 2 depicts the flow chart of our framework in detail for multiobjective 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, multiobjective performance assessment is presented using multiobjective metrics and approximate Pareto front.

## VI. 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 III.

TABLE I
Amazon EC2 Parameters Used in Our Experimental Study

| | Computing | | | | Networking | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Down | | | Up | | |
| | $\rho$ | $\epsilon$ | $C$ | $\beta$ | $\tau$ | $\epsilon$ | $\beta$ | $\tau$ | $\epsilon$ | $\beta$ |
| 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 |

### A. Pegasus Workflow and Amazon EC2

We evaluate the algorithms by utilizing workflows from Pegasus Workflow Management System [28] with various topologies from different domains. There are five different workflows, which are CyberShake, Montage, Inspiral, Epigenomics, and Sipht, where Fig. 3 symbolically represents their topological structures. We consider instances of Amazon Elastic Compute Cloud (Amazon EC2), which provides variety of resizeable capacity for CPU, memory storage, and networking in the cloud, tailored for different application profiles. Table I presents the computing capacity ($\rho$), network bandwith ($\tau$), cost ($C$), and reliability ($\beta$) of the instances.

Units are insignificant in most of the cases where each metric will be normalized into range of [0,1] for multiobjective metric calculations (see Section VI-B).

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 I illustrates network bandwidths in Gb/s.

We scale energy consumption by 1.8, which is not available to end users. We also select variety of $\beta$ 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.

In our experimental study, two physical fog network in tree topology is implemented using Amazon EC2 computing nodes [see the clusters in Fig. 4(a) and (b)]. We use *m5.24xlarge* instances on cloud side where abundant number of instances are available. Edge nodes will have instance of *m5.large* where

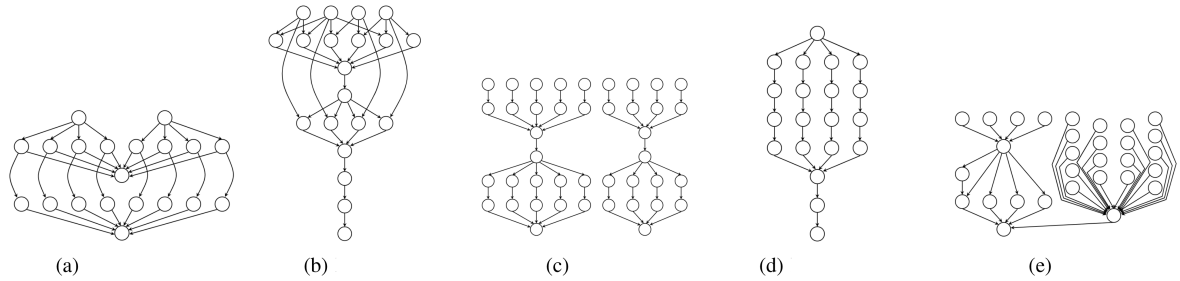


Fig. 3. Pegasus workflow. (a) Cybershake. (b) Montage. (c) Inspiral. (d) Epigenomics. (e) Sipht.

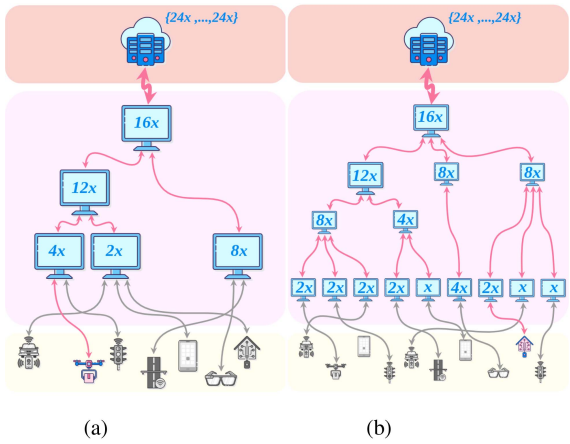


Fig. 4. Deployment schemes. (a) Cluster 1. (b) Cluster 2.

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 I ranging from *m5.xlarge* to *m5.16xlarge*.

### B. Metrics for Multiobjective Optimization

In order to asses performance of algorithms for MOPs, there are a number of metrics presented in the literature [29]. In our empirical evaluations, we select three widely used metrics in the literature, which are: unique nondominated front ratio (UNFR), $\Delta^*$, and the HV, respectively.

Since the true Pareto front is unknown for the experiments, we combine all nondominated 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.

*UNFR:* This metric indicates portion of nondominated solutions in the solution set. One of the issues about the commonly used version of this metric is that some multiobjective solvers may output duplicate solutions where duplicates would not dominate each other. Uniqueness in the nondominated set is injected into this metric to address this challenge [29]

$$\text{UNFR}(S) = \frac{|s \in S_{\text{unf}} | \nexists r \in R_{\text{unf}} : r \prec s|}{|R_{\text{unf}}|}. \tag{16}$$

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

$\Delta^*$: This metric measures the spread of the solutions in $S$

$$\Delta^*(S,P) = \frac{\sum_{k=1}^{m} d(\vec{e}_k, S) + \sum_{i=1}^{|S|} |d_i - \overline{d}|}{\sum_{k=1}^{m} d(\vec{e}_k, S) + (|S|)\overline{d}} \tag{17}$$

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 on $k$th objective. $\overline{d}$ is the average distance in the set $S$.

*HV:* It is one of the most commonly used metrics in the literature despite its high complexity [29]. HV 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 [30].

## VII. Experimental Results

The comparison study of the algorithms is 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 I. Thus, using five different CCR values from the {0.2,0.5,1,5,10} and the default CCR value given in [28], we conduct experimental study with 90 workflows. Metaheuristic algorithms are executed in the literature with fixed number of generations. On our preliminary experiments, 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 10 000. 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 sort 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 this article, execution times are equal for the reference algorithms unless otherwise stated.

We present three multiobjective metrics for 100 tasks of 5 workflows from Pegasus benchmark in Table II. The *MOMRank*

TABLE II
PERFORMANCE COMPARISON FOR MULTIOBJECTIVE METRICS IN SELECTED SET OF WORKFLOWS WITH NUMBER OF TASKS IS 100

| | | HV | | | | $\Delta*$ | | | | 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** |

The bold entities indicate the best performance of corresponding metrics on column header and benchmark instance on row headers.

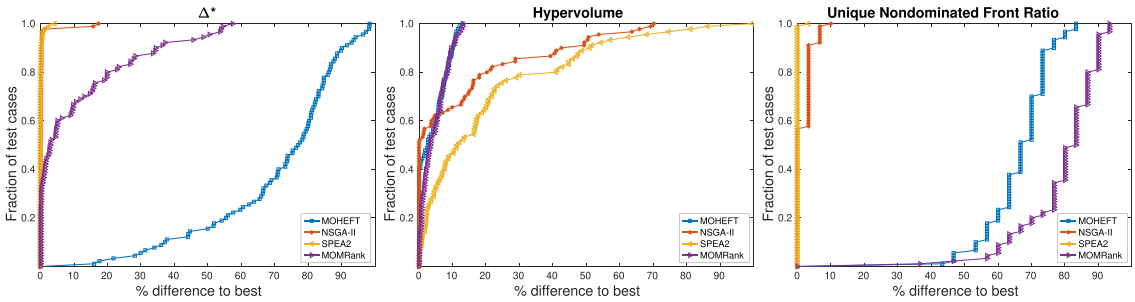


Fig. 5. Algorithm performances on both clusters for multiobjective metrics $\Delta*$, HV, and UNFR. (a)$\Delta*$. (b) HV. (c) UNFR.

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 [31], which helps to compare performance of algorithms in aggregated fashion over all problem instances. Fig. 5 depicts performance of the four algorithms (without considering enhancements of our algorithm on network congestion) on Cluster 1 in three metrics, $\Delta$* [see Fig. 5(a)], HV [see Fig. 5(b)], and UNFR [see Fig. 5(c)]. On this experiment [see Fig. 5], $x$-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.

## A. Performance Analysis on Multiobjective Metrics

The *MOMRank* algorithm reaches to the best spread value ($\Delta$*) for 33% of the test instances, which is due to the multiple task ranking approaches that cause diversity [see Fig. 5(a)] with respect to MOHEFT. On the other hand, the MOHEFT algorithm does not reach the best spread for any of the instances; it is 15% away from the best $\Delta$* for its best case. The NSGA-II and SPEA2 algorithms reach the best results in spread values ($\Delta$*), for almost every instances validating search space exploration feature of the meta-heuristics. For HV values [given in Fig. 5(b)], MOMRank and MOHEFT algorithms present almost the same best performance with 13% decline on worst case yielding smaller HV values. 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 Fig. 5(c)]. The SPEA2 algorithm outperforms other algorithms yielding most number of unique solutions for almost every instances.

## B. 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 V-A, 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 *CompMatching* algorithms. We separately illustrate ANC value further extending the objective set, which already comprises five different objectives (see Section III). ANC is calculated by the average time spent on data transmission over every physical network link [see Section III-B, (14)]. With the ANC metric, we ascribe higher prominence to network congestion for upcoming future internet beyond objectives in Section III-B. Fig. 6 depicts impact of DAG clustering approaches on

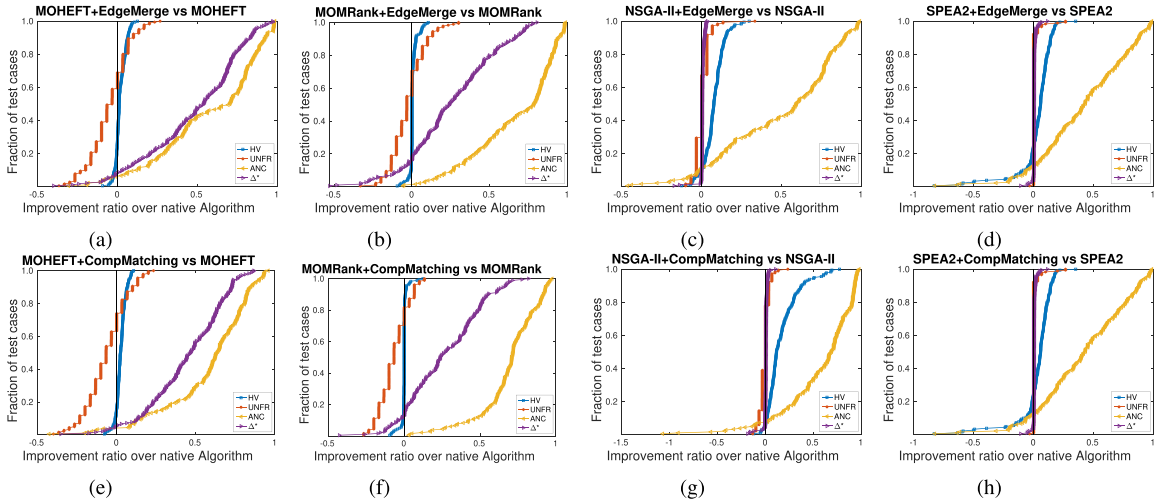


Fig. 6. Impact of DAG clustering extensions for multiobjective metrics and ANC on algorithms. (a) MOHEFT. (b) MOMRANK. (c) NSGA-II. (d) SPEA2. (e) MOHEFT. (f) MOMRANK. (g) NSGA-II. (h) SPEA2.

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.

The algorithms yield worse results for some multiobjective metrics like UNFR [see Fig. 6(a), (b), (e), and (f)]. 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 $\Delta^*$ 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 multiobjective metrics compared to the MOHEFT and the MOMRank algorithms. The HV values are improved for the SPEA2 and the NSGA-II algorithms for 60% and 80% of the instances on both extensions, respectively [see Fig. 6(d) and (h)].

Our final 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 III 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 III highlights that performance improvements for our task clustering extensions, *CompMatching* and *EdgeMerge*, on Fig. 6 are statistically significant. We can also conclude that NSGA-II extended with the proposed *CompMatching* clustering scheme, (NSGA-II+CompMatching) is the best performing algorithm in terms of HV metric.

TABLE III
STATISTICAL ANALYSIS OF ALGORITHMS BASED ON PAIR WISE COMPARISONS WITH RESPECT TO HV METRIC

| | MOHEFT | MOHEFT+CompMatching | MOHEFT+EdgeMerge | MOMRANK | MOMRANK+CompMatching | MOMRANK+EdgeMerge | NSGA-II | NSGA-II+CompMatching | NSGA-II+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+ |
| NSGA-II | s− | s− | s− | s− | s− | s− | *NA* | s− | s- | + | s− | − |
| NSGA-II+CompMatching | s+ | s+ | s+ | s+ | s+ | s+ | s+ | *NA* | s+ | s+ | s+ | s+ |
| NSGA-II+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* |

## VIII. CONCLUSION

In this study, we introduce a new multiobjective task scheduling algorithm for fog environments called the MOMRank algorithm and compare its performance in terms of a set of multiobjective metrics with three prominent scheduling algorithms. Algorithms in our empirical evaluation is extended with two task clustering algorithms to minimize data movement of workflows on the network clusters aiming to reduce network congestion. Our experimental study proves that added diversity with multiple task ranking to the scheduling algorithm can improve for set of multiobjective metrics and network congestion can be further minimized. 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.

## ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their detailed comments to further improve this work.

## 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] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.
[3] 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, pp. 1–41, Apr. 2019.
[4] 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 Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4590–4602, Oct. 2018.
[5] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Multi-objective task scheduling in cloud-fog computing using goal programming approach," *Cluster Comput.*, vol. 25, no. 1, pp. 141–165, Feb. 2022.
[6] T. Huang, W. Lin, C. Xiong, R. Pan, and J. Huang, "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5595–5608, Nov. 2021.
[7] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018.
[8] 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, pp. 1–43, Feb. 2022.
[9] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Inform.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018.
[10] L. Sanchez et al., "SmartSantander: IoT experimentation over a smart city testbed," *Comput. Netw.*, vol. 61, pp. 217–238, 2014.
[11] D. Zeng, 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 Trans. Comput.*, vol. 65, no. 12, pp. 3702–3712, Dec. 2016.
[12] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "DATS: Dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3423–3436, Apr. 2019.
[13] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2094–2106, Jun. 2018.
[14] Z. Xiao et al., "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 2038–2052, Mar. 2020.
[15] Q. He et al., "A game-theoretical approach for user allocation in edge computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 515–529, Mar. 2020.
[16] D. Zeng, L. Gu, and H. Yao, "Towards energy efficient service composition in green energy powered cyber-physical fog systems," *Future Gener. Comput. Syst.*, vol. 105, pp. 757–765, 2020.
[17] L. Gu, W. Zhang, Z. Wang, D. Zeng, and H. Jin, "Service management and energy scheduling toward low-carbon edge computing," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 1, pp. 109–119, Jan.–Mar. 2023.
[18] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved NSGA-II," *Wireless Pers. Commun.*, vol. 102, no. 2, pp. 1369–1385, 2018.
[19] M. A. Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Gener. Comput. Syst.*, vol. 124, pp. 142–154, 2021.
[20] N. Téllez, M. Jimeno, A. Salazar, and E. Nino-Ruiz, "A tabu search method for load balancing in fog computing," *Int. J. Artif. Intell.*, vol. 16, pp. 78–105, 2018.
[21] D. Ye, X. Wang, and J. Hou, "Balanced multi-access edge computing offloading strategy in the Internet of Things scenario," *Comput. Commun.*, vol. 194, pp. 399–410, 2022.
[22] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Inf. Sci.*, vol. 379, pp. 241–256, 2017.
[23] J. J. Durillo, H. M. Fard, and R. Prodan, "MOHEFT: A multi-objective list-based method for workflow scheduling," in *Proc. IEEE 4th Int. Conf. Cloud Comp. Technol. Sci. Proc.*, 2012, pp. 185–192.
[24] A. Iosup, M. Jan, O. Sonmez, and D. H. Epema, "On the dynamic resource availability in grids," in *Proc. IEEE/ACM 8th Int. Conf. Grid Comput.*, 2007, pp. 26–33.
[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[26] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," ETH Zurich, Comput. Eng. Netw. Lab., vol. 103, TIK Rep., 2001.
[27] J. Herrmann, J. Kho, B. Uçar, K. Kaya, and U. V. Catalyurek, "Acyclic partitioning of large directed acyclic graphs," in *Proc. IEEE/ACM 17th Int. Symp. Cluster, Cloud Grid Comput.*, 2017, pp. 371–380.
[28] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large-Scale Sci.*, 2008, pp. 1–10.
[29] S. Jiang, Y. Ong, J. Zhang, and L. Feng, "Consistencies and contradictions of performance metrics in multiobjective optimization," *IEEE Trans. Cybern.*, vol. 44, no. 12, pp. 2391–2404, Dec. 2014.
[30] C. M. Fonseca, L. Paquete, and M. Lopez-Ibanez, "An improved dimension-sweep algorithm for the hypervolume indicator," in *Proc. IEEE Int. Conf. Evol. Comput.*, 2006, pp. 1157–1163.
[31] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, pp. 201–213, Jan. 2002.

**Lokman Altin** received the B.S. and M.S. degrees in computer engineering from Marmara University, Istanbul, Turkey, in 2012 and 2015, respectively. He is currently working toward the Ph.D. degree in computer engineering with Boğaziçi University, Istanbul.

From 2013 to 2022, he was a Research Assistant at the Computer Engineering Department, Marmara University. He is currently working as a Software Engineer with Siemens Advanta, Istanbul. His research interest includes task scheduling in fog computing, multiobjective optimization, and evolutionary dynamic optimization.

**Haluk Rahmi Topcuoglu** (Member, IEEE) received the B.S. and M.S. degrees in computer engineering from Bogazici University, Istanbul, Turkey, in 1991 and 1993, respectively, and the Ph.D. degree in computer science from Syracuse University, Syracuse, NY, USA, in 1999.

He is currently a Professor with Computer Engineering Department, Marmara University, Istanbul. His research interests mainly include workflow scheduling in fog computing and cloud computing, task scheduling and mapping for multicore architectures, software-based hardware reliability, soft error propagation, and meta-heuristic based techniques for dynamic optimization problems.

Dr. Topcuoglu is a member of the IEEE Computer Society and the ACM.

**Fikret Sadik Gürgen** received the B.S. and M.S. degrees from the Electronical Engineering Department, Istanbul Technical University, Istanbul, Turkey, in 1980 and 1982, respectively, and the M.S. degree in computer engineering from Ohio State University, Columbus, OH, USA, in 1986, and the Ph.D. degree in electrical engineering from the University of Akron, Akron, OH, USA, in 1989.

He was in NTT, Tokyo, Japan, from 1989 to 1991. He joined the Computer Engineering Department, Boğaziçi University, Istanbul, in 1991. He was a Visiting Scientist at Sydney University Australia, Camperdown NSW, Australia, between 1993 and 1994. He was in Motorola Australian Research Center from 1995–1996. He was also a Visiting Scientist at Macquarie University, Sydney, NSW, Australia, in 1996. He became a Professor of Computer Engineering Department, Boğaziçi University, in 1998. He currently works in the same Department. He is the author of more than 100 publications.