An Automated Task Scheduling Model using Non-Dominated Sorting Genetic Algorithm II for Fog-Cloud Systems

Ismail M. Ali, Karam M. Sallam, Nour Moustafa, Ripon Chakraborty, Michael Ryan and Kim-Kwang Raymond Choo

***Abstract*—Processing data from Internet of Things (IoT) applications at the cloud centers has known limitations relating to latency, task scheduling and load balancing. Hence, there have been a shift towards adopting fog computing as a complementary paradigm to cloud systems. In this paper, we first propose a multi-objective task-scheduling optimization problem that minimizes both the makespans and total costs in a fog-cloud environment. Then, we suggest an optimization model based on a Discrete Non-dominated Sorting Genetic Algorithm II (DNSGA-II) to deal with the discrete multi-objective task-scheduling problem and to automatically allocate tasks that should be executed either on fog or cloud nodes. The NSGA-II algorithm is adapted to discretize crossover and mutation evolutionary operators, rather than using continuous operators that require high computational resources and not able to allocate proper computing nodes. In our model, the communications between the fog and cloud tiers are formulated as a multi-objective function to optimize the execution of tasks. The proposed model allocates computing resources that would effectively run on either the fog or cloud nodes. Moreover, it efficiently organizes the distribution of workloads through various computing resources at the fog. Several experiments are conducted to determine the performance of the proposed model compared with a continuous NSGA-II (CNSGA-II) algorithm and four peer mechanisms. The outcomes demonstrate that the model is capable of achieving dynamic task scheduling with minimizing the total execution times (i.e. makespans) and costs in fog-cloud environments.**

***Index Terms*—Genetic Algorithm, NSGAII, Task-Scheduling, Resource Allocation, Fog Computing, Cloud Computing**

I. Introduction

CLOUD computing is widely used by both public and private sector organizations, although a number of challenges associated with cloud services exist (e.g., lack of location-awareness, and latency). Hence, there have been attempts to complement existing cloud services using fog computing, particularly for latency- and delay-sensitive applications (e.g., battlefield Internet of Things). The fog is an architectural design, which extends the main services of the cloud to be deployed at the edge of a network. Specifically, fog is a considerably virtualized system of resource pools with a decentralized infrastructure, where computing resources can be shared between clients and cloud data centers to improve computational processing and data analytical capabilities [1], [2].

By executing data collection tasks at the edge of a network, task-related data analytics can be promptly undertaken at the fog, which address limitations such as latency and limited bandwidth. Computing and infrastructure resources, such as available nodes, computational processing, bandwidth, costs, and power, can also be optimally allocated to run tasks either in the fog or cloud environments, depending on the configurations and requirements. Task scheduling, along with fast and resilient processing and load-balancing, are crucial in the visualization technology to minimize costs and overheads (e.g., execution times) for both users and providers [3].

This reinforces the importance of having efficient scheduling and distributing of tasks between fog and cloud nodes [1], [2]. Cloud systems usually inter-operate using task scheduler and load balancer services that are not fully automated [2], [3]. The task scheduler allocates computing resources that can be executed at fog and/or cloud nodes, while the load balancer organizes the distribution of workloads through multiple computing resources. Task scheduling algorithms are then used to schedule tasks on computing processors for minimizing the total makespan without violating the given constraints [4]. Multiprocessor scheduling is categorized as an NP-hard problem, where the algorithms fundamentally apply single objective optimization functions that minimize execution times or costs [3], [4]. Fog-cloud systems generally comprise multiple computing nodes and high-performance servers with large number of processor cores [5].

However, as the interactions between fog and cloud nodes are relatively complex, a multi-objective optimization function is required. For example, allocating resources (e.g., processor cores, memory, and I/O interrupts) to handle user tasks can be challenging, particularly in dynamic fog-cloud environments [5], [6]. In shared-memory multiprocessor systems, the connections between central memory, I/O interrupts and processors require an optimization process to facilitate task executions with a load-balancing capability. Furthermore, different task executions have different resource constraints that demand regular updates to examine their status (e.g., available or still occupied). Optimization and meta-heuristic approaches have

Ismail M. Ali, Karam M. Sallam, Nour Moustafa, Ripon Chakraborty, Michael Ryan are with the School of Engineering and Information Technology, University of New South Wales at ADFA, Northcott Dr, Campbell ACT 2612, Canberra, Australiae-mail: ismail.ali@student.adfa.edu.au, k.sallam@unsw.edu.au,nour.moustafa@unsw.edu.au, r.chakrabortty@adfa.edu.au, m.ryan@adfa.edu.au .

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, University of Texas at San Antonio, San Antonio, TX 78249, USA, e-mail: raymond.choo@fulbrightmail.org

been widely used to facilitate task scheduling and resource allocation [7], [8], [9], [10]. However, existing approaches generally require manual tuning of parameters and have a number of limitations [11], [12], [13].

In this paper, we propose a multi-objective task scheduling problem to minimize the total makespans and costs in fog-cloud environments. One-point crossover is used to design a discrete NSGA-II (DNSGA-II) model, which directly specifies computing nodes, rather than the use of mapping methods that require high computational resources. The process of individual reproduction in one-point crossover depends on the production of a new offspring by swapping parts from two other individuals (called parents) so that the generated offspring has characteristics from its parents but is not similar to them. This way of individual reproduction can construct new discrete solutions that are able to further explore the search space and reduce the population similarity. This, therefore, guides the search process of DNSGA-II and improves its global optimization ability, especially in distributed environments such as those of fog-cloud systems.

The key contributions of this paper include the following:

- We formulate the task-scheduling problem as a multi-objective optimization problem for minimizing the total execution times and costs to dynamically allocate appropriate resources to predefined tasks in a fog-cloud environment.
- We propose an enhanced NSGA-II optimization model that can deal with discrete multi-objective optimization problems to assign particular computing nodes (e.g., nodes 1 and 2), which cannot be allocated using a continuous space. The proposed model is designed to adaptively tune extensive parameters to select their best values.
- We assess the performance of the proposed NSGA-II model for scheduling tasks and dynamically distributing them in fog-cloud environments. The model is tested to auto-allocate tasks with appropriate resources either in a fog tier, a fog-cloud tier or a cloud tier, and is compared with other compelling models.

The rest of this work is structured as follows. Section II explains the background and related work of the NSGA-II model and task scheduling in fog-cloud environments. The problem of task-scheduling is discussed in Section III. The proposed model and its fog-cloud architecture are then described in Section IV. The experimental results are provided in Section V and some conclusions are drawn in Section VI.

II. Background and Related Work

This section explains concepts and related studies to task scheduling and resource allocation in fog, fog-cloud and cloud environments, as well as discusses multi-objective optimization and NSGA-II approaches.

A. Task Scheduling and fog-cloud Systems

With the prevalence of IoT applications, there has been an increasing interest in moving computing and storage at the edge of a network close to users and organizations. In more detail, cloud data centers would not handle big data collected from those applications in real-time, due to their limitations of delay, limited bandwidth and geographical distributions [5], [11]. The cloud's limitations bring attention to move data processing and analytic models onto either fog computing or fog-cloud computing models for faster processing [5], [14], [15]. To enable faster processing for the fog/fog-cloud models, it is vital to develop effective task scheduling techniques.

Task scheduling in computing platforms is broadly classified into dependent and independent approaches [16]. In the dependent approaches, there are dependencies and communications while distributing and allocating tasks to particular computing resources. In the independent approaches, tasks are individually distributed on computing resources either using a batch mode or an online mode [16]. Both modes describe the arrival way of the tasks, where in the batch mode, the tasks are allocated to the corresponding fog nodes through a scheduling algorithm once they arrived, while in the online mode, the allocations of the arrived tasks are processed through a Resource Management System (RMS), as the arrival time of each task is random [16]. In our architecture, the online mode has been considered as it is more realistic and reflects the task arrival and allocation processes in the real world.

Task scheduling algorithms are essential in dynamic environments to allocate batch mode tasks and assist in organizing the arrival time of every task scheduled for the resource management system. Task scheduling has been formulated as an NP-complete problem [17]. Despite the advantages of fog-cloud systems, task scheduling and resource allocation in such systems still suffer from challenges related to dynamic environments and auto-configuration of tasks and their required resources [10]. The task scheduling problem in dynamic environments, such as fog-cloud systems, demands the development of optimization models that could automatically tune its parameters and specify appropriate computing resources to either fog or cloud nodes [12], [14], [18]. There is also a lack of design for a fog-enabled cloud architecture that demonstrates the communication between fog and cloud elements and their resource distributions [16]. ,

Optimization and meta-heuristic algorithms have been widely employed to schedule tasks and allocate batch or online modes [5], [10]. For example, Naeem et al. [19] stated that 46,000 IoT units were installed between 2016 and 2020, where heterogeneous resources and dynamic workloads needed for each IoT application introduce new issues in fog and fog-cloud systems. In light of these open challenges, there have been numerous attempts by researchers to allocate tasks in fog and fog-cloud environment. Pham et al. [20] considered a task scheduling problem, while they proposed a cost-makespan aware scheduling approach to obtain the highest cost-makespan trade-off value. They prioritized each node before allocating specific tasks to them. Although their approach hinted at the necessity of cost-makespan trade-off, they considered a rational factor to linearize that trade-off rather than using any Pareto optimization concepts.

In another work, Zeng et al. [21] concurrently considered task image placement and task scheduling strategies to minimize the completion time of service requests. By balancing the workload task on both client and fog nodes, they attempted

to minimize the transmission latency of all requests. A three-stage heuristic algorithm was proposed to their mixed-integer nonlinear programming model. A method proposed by Xu et al. [22] conveyed a laxity-based priority algorithm to allocate tasks on fog-cloud nodes. They attempted to minimize total energy consumption during the task and resource allocation and proposed a constrained optimization model which was later solved by an Ant Colony Optimization (ACO) algorithm. However, in their proposed model, they did not consider independent and associated tasks for scheduling. Moreover, their proposed approach was only validated against small datasets.

A similar study was also carried out by Boveiri et al. [23] who proposed a variant of ACO algorithm, the so-called max-min ant system, for task scheduling in cloud applications. Different priority values of tasks were considered to select the optimal combinations of task-orders. After exhaustive experimental analysis on randomly generated cloud-based computing data, they proved the suitability of their algorithm against other counterparts. However, their model was limited to small-sized task-graph input samples in the cloud only. Recently, Nguyen et al [10] also considered a fog-cloud environment and proposed a time-cost aware scheduling algorithm to allocate tasks in different nodes. Although their approach is very useful to keep track of both cost and makespan associated with a resource and task allocation, as with Pham et al. [20], they did not consider any Pareto optimization algorithms to intelligently handle two-conflicting objectives (e.g., minimizing both costs and makespan).

From the discussions above, most of the existing task scheduling approaches are only valid for small datasets and only considered either cost or makespan (i.e., total completion time) as a single objective. Although some trade-off analysis has been conducted in the existing works, such trade-offs could not reflect reality. To deal with two or multiple contrasting objectives, Pareto optimization techniques, or multi-optimization algorithms are essential. Jena [24] proposed a nested-PSO algorithm to solve a multi-objective task scheduling problem. Their objectives were to minimize energy and processing time. However, their analysis was only limited to the cloud computing environment. More recently, Kumar and Venkatesan [25] proposed one multi-objective task scheduling methodology, solved by a hybrid genetic-ACO algorithm.

Most of existing task scheduling techniques has been widely formulated as a multi-objective problem, which is more realistic than a single objective problem [26], [27], [28]. The multi-objective problem makes a trade-off between the objectives, including response time vs. computation cost or power consumption vs. computation latency, should be optimized to make the best decision(s). Consequently, this study proposes a multi-objective optimization problem that deals with task-scheduling problems in both cloud and fog-cloud environments and discusses the computation differences between them. To do so, we develop a modified NSGA-II algorithm that can efficiently handle multi-objective optimization problems.

TABLE I
NOTATIONS OF MATHEMATICAL SYMBOLS USED IN THE STUDY

Symbol	Description
P_i	Server number i
T_j	Task number j
M	Total number of available servers
N	Total number of tasks
T_i^j	Task j is processed by server i
$I(T_j)$	Total number of j task instructions
$RM(T_j)$	Required memory for task j
$RB(T_j)$	Required bandwidth to upload and download files of task j
$S_r(P_i)$	Computing rate of server i to handle million instructions per second
$S_{usage}(P_i)$	Computing cost of server i
$Mem_c(P_i)$	Cost of memory usage of server i
$BW_c(P_i)$	Cost of bandwidth usage of server i
P_{jeoi}	Node (i) and storage (o) servers have been assigned a task (j) that is requested by client (e) to be processed
E	Total number of clients

B. NSGA-II Algorithm

NSGA-II is a powerful algorithm for solving Evolutionary Multi-Objective (EMO) problems. The main characteristics of NSGA-II include: 1) a fast sort method of non-dominated solutions, 2) a fast estimation technique of the density of solutions, and 3) a simply crowded selection strategy [29]. NSGA was basically designed for solving multiple objective optimization problems with continuous decision variables. However, a binary representation of solutions has also been used for applying classical genetic operators, such as a one-point crossover, a point mutation [7], logical crossover and mutation operators [30]. It was recommended by [7] the use of a real-valued representation with specific genetic operators, such as Simulated Binary Crossover (SBX) and polynomial mutation [31], for solving continuous function optimization problems.

Since multi-objective optimization problems involve more than one objective function to be optimized simultaneously, and with the absence of any additional information about the EMO problem to be solved, no particular Pareto-optimal solution can be considered as better than the others [32]. Therefore, the optimal solutions (decisions) need to be determined as a trade-off between two or more objectives and the optimality of a solution varies based on a number of factors, such as user's choice, problem definition or its environment. Considering the merits of NSGA-II, its implementation in the cloud, fog and fog-cloud environment would be an interesting research work. However, since NSGA-II was originally developed for solving continuous problems, it needs to be modified to solve discrete ones, as we propose in this study.

III. FORMULATION OF TASK SCHEDULING PROBLEM IN FOG-CLOUD SYSTEMS

In fog-cloud systems, the fog layer receives several requests from many IoT applications. The requests are then decomposed into small independent tasks, characterized by many factors, for example, the number of instructions, memory required, as well as input and output file sizes. The tasks have to be processed using the resources of the fog-cloud systems, which comprise computing servers with various ca-

pabilities. The servers could be located at either fog or cloud nodes, where each of these has different resources that enable computing, including core processors (e.g., CPU), memory bandwidth, storage, and I/O usage. It is observed that the costs of using cloud nodes are higher, but such nodes are more powerful than their fog counterparts [2].

The notations of most of the mathematical symbols are listed in Table I. In a typical task scheduling problem, the objective is to schedule the entire tasks to minimize the total execution times (i.e., makespans) using the minimum total costs of available resources. For a single request, let N be the number of tasks ($T = \{T_1, T_2, ..., T_j, ..., T_N\}$) to be processed, M the number of available servers ($P = \{P_1, P_2, ..., P_i, ..., P_M\}$). For task T_j, let $I(T_j)$ be the number of instructions, $RM(T_j)$ the required memory, and $RB(T_j)$ the required bandwidth to upload and download T_j's files.

For each server (P_i), let $S_r(P_i)$ be the computing rate, which is the server capability to compute million instructions per second, $S_{usage}(P_i)$ the computing cost of server/node (P_i), $Mem_c(P_i)$ the cost of memory usage, and $BW_c(P_i)$ the cost of bandwidth usage. Mathematically speaking, the set of tasks (T) and processors (P) can be presented using the following vectors: $T = \{T_1, T_2, T_3, ..., T_N\}$, $P = \{P_1, P_2, P_3, ..., P_M\}$. Also, the solution can be presented as follows, where T_j^i denotes that task j is processed by server i:

$$Sol = \{T_1^i, T_2^i, T_3^i, ..., T_j^i, ..., T_N^M\} \tag{1}$$

The task scheduling problem can be formulated as a multi-objective optimization problem by using the following mathematical model:

$$Minimize: \sum_{i=1}^{M}(max_{1\leq j\leq N}(\frac{I(T_j)}{S_r(P_i)}),\ \forall\, T_j \in P_i) \tag{2}$$

$$Minimize: \sum_{i=1}^{M}(\sum_{T_j\in P_i}(TotalCost(T_j^i))) \tag{3}$$

Subject to

$$\sum_{o\in O}\sum_{i\in P_M} p_{jeoi} + \sum_{o\in O} p_{jeo} = 1,\ \forall j \in T_N, e \in E \tag{4}$$

$$y_{jo} = \begin{cases} 1, & if\ Task\ j\ stored\ in\ node\ i \\ 0, & otherwise \end{cases} \tag{5}$$

$$\sum_{o\in O} y_{jo} = \Omega,\ \forall j \in T_N \tag{6}$$

$$\sum_{i=1}^{M} T_j^i = 1,\ \forall\, T_j \in T,\ \forall\, Pi \in P \tag{7}$$

and

$$i > 0, \quad i = 1, 2, ..., M \tag{8}$$

$$j > 0, \quad j = 1, 2, ..., N \tag{9}$$

$$o > 0, \quad o = 1, 2, ..., O \tag{10}$$

where $\frac{I(T_j)}{S_r(P_i)}$ is the execution time required by node i to process task j, which can be obtained by dividing the number of instructions of task j (I) to the processing unit's rate of node i (S_r), $TotalCost(T_j^i)$ is the total computing cost of P_i, memory and bandwidth resources used by node i to process task j and Ω is a variable that needs to be considered and maintained for any task t on different servers m for system reliability and load balancing. p_{jeoi} indicates that node (i) and storage (o) servers have been assigned a task (j) that is requested by client (e) to be processed and E is the total number of clients. $TotalCost(T_j^i)$ can be calculated as the sum of the three combinations, as follows:

$$TotalCost(T_j^i) = S_c(T_j^i) + M_c(T_j^i) + B_c(T_j^i) \tag{11}$$

Every cost can be independently calculated using the following three equations. The equations present the processing, RAM usage and bandwidth usage costs, respectively, required by the node i to process the task j.

$$S_c(T_j^i) = S_{usage}(P_i) \times (\frac{I(T_j)}{S_r(P_i)}) \tag{12}$$

$$M_c(T_j^i) = Mem_c(P_i) \times RM(T_j) \tag{13}$$

$$B_c(T_j^i) = BW_c(P_i) \times RB(T_j) \tag{14}$$

In the model, equation (2) presents the first objective function, which minimizes the total response time required by a system (i.e. fog-cloud) to handle users' requests. Equation (3) shows the second objective function, which minimizes the total cost of the resource usages. The first constraint (equation 4) ensures that all the tasks ($j \in T_N$) that are requested by e client are processed by either cloud or fog nodes to ensure the task completeness. The second constrain (equations 5) checks the existence of j task in a storage server by using a binary variable y_{jo}, which decides whether a storage server (o) holds a task j or not. While the third constraint (equation 6) puts a limitation on the number of stored tasks on a particular storage server. The fourth constraint (equation 7) ensures that task j is processed by only one node. The last three constraints (equations 8, 9 and 10) preserve the discrete domain of tasks, computation servers and storage servers, respectively.

IV. Proposed Task scheduling Model and its Architectural Design

A. Proposed Task scheduling Model

This section explains the proposed task scheduling model using an enhanced NSGA-II algorithm. The NSGA-II algorithm fundamentally relies on continuous crossover operators. The operators do not suit a task scheduling problem in dynamic fog-cloud environments because continuous values resulting from the Continuous NSGA-II (CNSGA-II) algorithm can not be assigned appropriate computing nodes. For example, if the output of the algorithm is a node vector of $2.5, 3.7, 5.8$, each value in the vector has to be mapped into discrete values of $2, 3, 5$ that refer to the computing nodes required, either in the fog or the cloud tier. This problem has been solved using mapping functions, which sometimes demand high computational resources, especially in the dynamic environments that have broader ranges of resources.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3032386, IEEE Transactions on Cloud Computing

In order to address this problem, we directly employ a Discrete NSGA-II algorithm (DNSGA-II). This model has practically proven its high performance, in terms of processing time and costs, compared with CNSGA-II. The DNSGA-II algorithm is also enhanced to auto-tune its parameters without human intervention; hence this enhanced version is utilized to address the automatic task distributions and allocate their proper computing resources in fog-cloud systems. The aim of the proposed task scheduling-based DNSGA-II is to optimize two objectives: minimum execution times (i.e., makespans) and minimum costs. These objectives are constrained by that every task has to processed using enough resources at fog or cloud nodes, as mathematically formulated in Section III. The main components of the task scheduling-based DNSGA-II are presented in Algorithm 1. The components of the model are elaborated below.

B. Solution representation and initial population

Initially, every solution in the population is represented by a vector of integer values in the range [1 - M], where M is the total available nodes/processors and the length of each vector equals the number of tasks to be scheduled (N). The integer value in each gene (T_j^i) represents the task (T_j) the will be handled by the processor (P_i). The number of the generated solutions equals the population size (PS). For each solution in the initial population, each task (T_j) is randomly assigned to the processor (P_i). For example, if we have 7 tasks (N= 7) to be processed by 3 processors (M= 3), one candidate solution can be represented as in Fig.1.

C. Fitness evaluation

In order to decide which solutions should be kept for the next generation and which should be excluded from the population, the quality of all solutions needs to be measured. In this work, the fitness values of the candidate solutions are evaluated against two objective functions: 1) makespan, which is the total time that is needed by the nodes to execute all the received tasks, and 2) total cost, which is the total resources, such as RAM, CPU, network bandwidth, required by the nodes to process the tasks. Both objective functions are formulated by equations 2 and 3.

D. Sorting of population

A fast-non-dominated sorting mechanism [33] is applied to sort solutions within the population. In this procedure, each solution (S_i) has an integer value that is holding the number of solutions, which dominate S_i. This holder integer value called "domination count". Another set, which contains all solutions that are dominated by the solution S_i is calculated. Each solution in the population is assigned a rank, based on those two parameters, which represents the front to which this solution belongs with rank equal to 0 for the Pareto front solutions. In general, all solutions, which have rank x+1 are dominated by solutions with rank x. For example, the solutions, which are dominated by solutions from the Pareto front, have rank 1.

Algorithm 1 Pseudo-code of Proposed task scheduling-based DNSGA-II

1: $Pop \leftarrow InitialPop(PS, N)$;
2: $[makespan, cost] \leftarrow FitnessEvaluation(Pop)$; (sub-section IV-C)
3: $[Pop_{sorted}] \leftarrow fastNonDominatedSort(Pop)$; (sub-section IV-D)
4: $Gen \leftarrow 1$;
5: **while** $Gen \leq Max\_Gen$ **do**
6: $Selected\_vectors \leftarrow RankSelection(Pop_{sorted}, PS)$; (sub-section IV-E)
7: $Trail\_vectors \leftarrow Crossover(Selected\_vectors, P_c)$; (sub-section IV-F)
8: $Mutant\_vectors \leftarrow Mutation(Trail\_vectors, P_m)$; (sub-section IV-G)
9: $Pop_{new} \leftarrow Mutant\_vectors$
10: $[makespan, cost] \leftarrow FitnessReEvaluation(Pop_{new})$;
11: $Pop_{union} \leftarrow Merge(Pop, Pop_{new})$; (sub-section IV-H)
12: $Fronts \leftarrow fastNonDominatedSort(Pop_{union})$;
13: $X \leftarrow [\,]$; // to keep Parents vectors
14: $F_L \leftarrow [\,]$; // to keep Front solutions
15: **for** $F_i \in Fronts$ **do**
16: $CrowdingDisAssignment(F_i)$;
17: **if** $Size(X) + Size(F_i) > PS$ **then**
18: $F_L \leftarrow F_i$;
19: Break();
20: **else**
21: $X \leftarrow Merge(X, F_i)$;
22: **end if**
23: **end for**
24: **if** $Size(X) < PS$ **then**
25: $F_L \leftarrow RankAndDistanceSorting(F_L)$;
26: **for** $j = 1 : PS - Size(Front_L)$ **do**
27: $X \leftarrow X_j$;
28: **end for**
29: **end if**
30: $Selected\_vectors \leftarrow RankAndDisSelection(X, PS)$;
31: $Pop \leftarrow Selected\_vectors$;
32: $Gen \leftarrow Gen + 1$;
33: **end while**

E. Selection operator

The first step in the main loop of GA is the selection operation. In the proposed NSGA-II, binary tournament selection [34] is used. In this approach, two solutions (parents) are selected from the population pool based on the rank for the reproduction process by crossover and mutation operators. A solution is selected if and only if its rank is lower than the other. Solutions that have the same non-dominated rank in the population are assigned a diversity rank by adapting an explicit diversity-preservation or niching strategy [12]. Genes in each non-dominated individual are ranked based on the density of their region. The density of solutions around a specific individual can be calculated using a crowding distance metric, which is the average distance of the two solutions on either

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3032386, IEEE Transactions on Cloud Computing

side of that solution along each of the objectives. Because such niching strategy does not require external parameters, it is used by NSGA-II [35].

F. Crossover operator

Crossover is considered the key operator for solution exploration in a GA [36]. New offspring (modified solutions) are reproduced by performing crossover and/or mutation operations. Generally in a crossover, two or more selected parents produce one or more offspring. In this paper, for the purpose of further studying the performance of the NSGA-II with continuous and discrete evolving operators, two different crossover operators will be used: 1) one-point crossover, and 2) a simulated binary crossover (SBX).

In one-point crossover, two solutions are randomly selected as parents, a crossover point (CP) is chosen uniformly at random between 1 and the solution length (equation 15), and two new solutions are created for the two parents. This CP divides each solution into two sub-solutions (left and right) and then the right (or left) sub-solution of the two individuals are swapped. For example, Fig. 1 considers the individual in sub-section IV-B as one parent (P_1) and another one is P_2. OS_1 and OS_2 are two offspring, which are produced from recombination of P_1 and P_2, with the CP occurs after the fourth bit.

$$CP = round((\alpha_{max} - \alpha_{min}) \times rand + \alpha_{min} \tag{15}$$

where α_{max} and α_{min} are the upper and lower limits of CP, $rand$ is a uniform random number generated in the range [0,1] and $round$ is a function applied to round the produced real number to its closest discrete one.

SBX crossover was firstly proposed by Deb and Agrawal [31] and it is used by the basic NSGA-II [29]. SBX is a real-parameter recombination operator, which is commonly used for solving optimization problems in continuous domain. It mimics the one-point crossover with binary coding, so it uses two solutions as parents and produces two offspring. Also, it uses a parameter called the distribution index (d_i), which is kept fixed throughout a simulation run. If d_i is large, the offspring is produced close to the parent, otherwise it will be produced away from the parent. d_i parameter has a direct effect in controlling the produced offspring distribution.

G. Mutation operator

Mutation is considered a key operator that increases the diversity of the population and enables GAs to explore promising areas of the search space [37]. Normally, after offspring are produced by crossover, mutation is applied to their variables according to a low probability called the mutation rate (p_m). It alters a few random bits of a solution and maintains genetic diversity, with the aim of preventing convergence towards a local optimum. For the purposes of studying the performance of NSGA-II using both continuous and discrete evolving operators, polynomial and a discrete bit-flip mutations are used.

In continuous NSGA-II (CNSGA-II), the polynomial mutation [38] is used. This mutation is based on the polynomial distribution and applied with a user-defined index parameter (Pl_m) to perturb the current solution (parent) to a neighbouring solution (offspring) by mutating each solution separately, i.e. one parent solution gives one offspring. The two main steps of polynomial mutation are as follows: 1) Calculate the perturbation parameter (Pl_m) by using the following rule,

$$Pl_m = \begin{cases} (2 \times rand)^{\frac{1}{r+1}} - 1 & , \ rand < 0.5 \\ 1 - [2(1 - rand)]^{\frac{1}{r+1}} & , \ rand \geq 0.5 \end{cases} \tag{16}$$

where $rand$ is a random number generated in the range [0-1] and r a positive or negative exponent decided by the user. 2) The new offspring is then produced from the original solution as follows:

$$OS = P + Pl_m \times \delta \tag{17}$$

where P is the parent solution, δ the maximum perturbation allowed between the original and mutant solution, which is also decided by the user.

In our algorithm and based on [38] and after doing some experiments for validating them, r is set to 100 and δ is calculated as

$$\delta = min(\frac{(OS - X_l), (X_u - OS)}{X_u - X_l}) \tag{18}$$

where X_l and X_u are the lower and upper values of solutions.

The produced offspring (OS) usually contains real-values. In order to use this mutation to the discrete task scheduling problem, the produced continuous mutant vectors have been rounded to their nearest discrete values. In the discrete NSGA-II (DNSGA-II), the traditional bit-flip mutation has been modified to produce discrete variables instead of binary ones and employed. The pseudo-code of bit-flip mutation is presented in Algorithm 2.

Algorithm 2 Pseudo-code for the bit-flip mutation

```
M ← size(child) // child is the vector produced from Crossover;
for k= 1 : M do
    R ← rand(); // a uniform random number generated in the range [0,1]
    if R < P_m then
        child(k) = randi(M); // randi is a discrete uniform random number generated in the range [1,M]
    end if
end for
```

H. Fitness re-evaluation and selection

The fitness values of the produced solutions in the new population (Pop_{new}) are re-evaluated. Then, both old (Pop) and new (Pop_{new}) populations are combined in one population (Pop_{union}) of size $2 \times PS$. The individuals of Pop_{union} are then sorted into a hierarchy of sub-populations based on the ordering of Pareto dominance. The similarity between individuals of each group is measured in the Pareto front, and the resulting groups and similarity values are used to endorse a diverse front of non-dominated solutions. The non-dominated solutions are emphasized and the Pareto front individuals are formed as Pareto-optimal solutions [9]. The pseudo-code of the

fitness re-evaluation and selection are illus tarted in Algorithm 1 through steps [12-30]. Finally, the new population is formed by selecting all front solutions with rank 0, followed by solutions with higher ranks.

I. Termination conditions

The processes of the NSGA-II operations continue until the allowed time or number of generations (Gen) exceeds a pre-defined maximum number of generations (Max_Gen). Also, the NSGA-II algorithm termination can be decided by the maximum allowed number of evaluations of the fitness (objective) functions.

J. Overall Model Architecture

The proposed task-scheduling model is designed in a fog-cloud environment, as depicted in Figure 1. The process of task-scheduling starts when a client requests the implementation of a particular task, such as analytics of big data sources collected at the edge layer. Then, the proposed model could be executed at the edge of a network using virtualization technology. The technology would enable the model to allocate the demanded resources at either the fog or the cloud layer. The selection of which layer is better depends on the costs and time executions of tasks, as detailed in Section V. The model executes its procedure, including initialization, discrete crossover and mutation operations to dynamically allocate tasks of applications to fog and/or cloud nodes. The model could also activate the load balancer service via the regular examination of the status of fog and cloud nodes (e.g., idle, occupied).

The main components of the fog-cloud system architecture include three layers of the edge, fog, and cloud, as shown in Figure 1. The edge layer includes the proposed task-scheduling model deployed in a virtualized server and linked with virtualized fog nodes and cloud centers using network gateways. The fog layer consists of a set f_M of fog nodes/computation servers and f_O of storage servers. In this layer, computing processes occur at devices that are close to end-users, such as local servers, gateways, etc. The cloud layer contains a set C_M of cloud computing servers and C_O of storage servers. All the servers in both Cloud and fog layers are interconnected with each other and with the clients. The fog layer contains a hierarchical computing architecture, where the proposed task scheduling model distributes the tasks/workloads to be parallelly processed by available computing nodes according to the length/complexity of each task and the maximum capacity of each node [39]. The proposed model would be also deployed at a fog mediator, which is a physical or virtual server that receives the requests from a set E of clients. Then, it distributes the tasks as independent ones to be processed on various fog-cloud nodes.

When a client request (r) sends to a (e) fog mediator, along with an average arrival rate (Λ_{re}). The proposed task scheduling based on DNSGA-II starts the execution process of decomposing the request into several independent tasks, which are then stored in a storage device (o) with capacity (K_o). A storage device is also used to handle any I/O interrupts, which could accidentally occur during task processing on a server/node. In this system, the I/O interrupts probability is assumed to be very small $\alpha = 0.0001$. After that, the tasks are transmitted to the computation server, which was assigned by the proposed task scheduler, to be executed. The transmission time can be vary depending on the environment. Although the computing cloud servers usually have a higher processing rate than fog servers, transmission latency in the cloud environment is larger than the fog environment. As the task execution time (makespan) could be impacted by this latency, the proposed model has a critical role in balancing the distribution of tasks into fog and cloud computing servers.

V. Experimental Results and Comparisons

The experiments presented in this study were conducted in a PC with Windows 10, a 3.4 GHz Core I7 processor and 16 GB RAM. The proposed algorithm is coded in Matlab 2018b. To judge the performance of the proposed NSGA-II, we solved 33 different task-scheduling problems (11 for fog, 11 for fog-cloud and 11 for cloud) with a different number of tasks (40 to 500 tasks).These problems are randomly generated the attributes listed in Table II. A different number of fog and cloud nodes has been with various characteristics as presented in Table III. The generated data instances and source codes of this proposed model can be accessed at [40] for future research directions and comparison purposes.

TABLE II
ATTRIBUTES USED TO GENERATE TASKS

Property	Value	Unit
Size of Input file	$[1, 100]$	10^9 instructions
Size of output file	$[50, 200]$	Mega Byte (MB)
No. of instructions	$[10, 100]$	Mega Byte (MB)
Required Memory	$[10, 100]$	Mega Byte (MB)

TABLE III
SUMMARY OF CHARACTERISTICS OF THE FOG-CLOUD STRUCTURE

Parameter	fog tier	Cloud tier	Unit
Bandwidth usage cost	$[0.01, 0.02]$	$[0.05, 0.1]$	Grid $ per MB
Memory usage cost	$[0.01, 0.03]$	$[0.02, 0.05]$	Grid $ per MB
CPU rate	$[500, 1500]$	$[3000, 5000]$	Million Instructions per Second
CPU usage cost	$[0.1, 0.4]$	$[0.7, 1.0]$	Grid $ per Second

The Wilcoxon rank-sum test with $\alpha = 0.05$ was used to check the statistical difference between the compared algorithms, with $+$, $-$, and $\approx$, which denote that the obtained results from the first algorithm are statistically better than, worse than, and similar to the obtained result by another algorithm, respectively. The efficiency of the proposed algorithm is also graphically assessed by plotting performance profiles [41], [42]. The performance profiles are drawn to compare the efficiency of different algorithms (A) using a number of problems (P) and a comparison goal (i.e., the computational time or the average number of FES) to attain a certain level of efficiency value (i.e., optimal objective function value). To estimate the performance profile Rho_a for an algorithm (a), the following equation is employed as:

$$Rho_a(\tau) = \frac{1}{n_p} \times |p \in P : r_{pa} \leq \tau| \tag{19}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3032386, IEEE Transactions on Cloud Computing

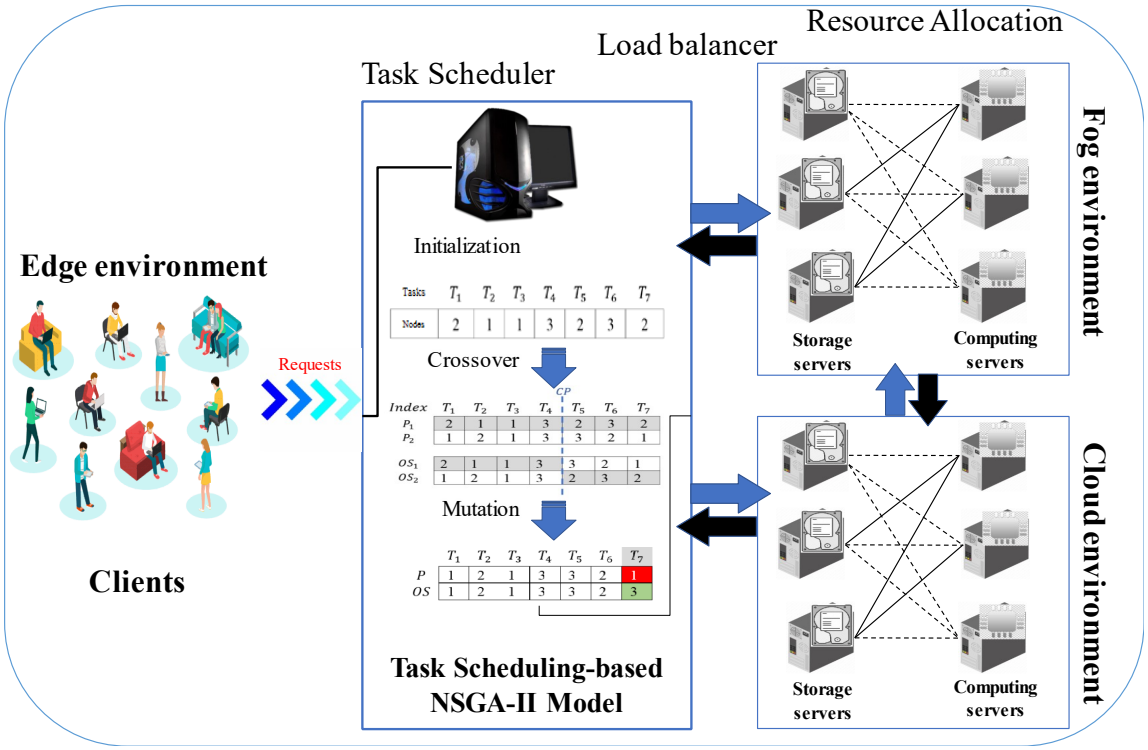


Fig. 1. Proposed task scheduling-based NSGA-II Model in fog-cloud environments

where $Rho_a(\tau)$ is a probability of $a \in A$ that the efficiency ratio $r_{p,a}$, which is computed by equation 20, is within a factor $\tau > 1$ of the best possible ratio and Rho_a is the cumulative distribution function of the performance ratio.

$$r_{pa} = \frac{t_{pa}}{min\{t_{pa} : a \in A\}} \tag{20}$$

where t_{pa} is the consumed CPU time by an algorithm a to attain the objective function value fit_p in problem p.

A. Parameter Tuning

The proposed NSGA-II algorithm has three main parameters: population size (PS), crossover rate (P_c) and mutation rate (P_m) that would affect the quality of the obtained results. Table IV presents the proposed values of the parameters. In order to determine the best combination of the parameter values, we use the Taguchi design method, which is a favorite method using orthogonal arrays. For every combination of the orthogonal array, 3 instances were randomly chosen from 11 ones to calculate the average total cost and average makespan values.

TABLE IV
COMBINATION OF PARAMETER VALUES

Parameters	Factors				
	1	2	3	4	5
PS	25	50	75	100	150
P_c	0.25	0.45	0.65	0.85	1.00
P_m	0.01	0.05	0.10	0.15	0.20

The proposed NSGA-II was run 30 runs, for each run, the best obtained cost and makespan were recorded. The numerical results of each parameter combinations are demonstrated in Table V. The values of each parameter are plotted in Figure 2 to adopt the best one. It can be concluded from Figure 2 for both average total makespan and average total cost that the largest values are better for both PS and P_c, while the lowest value is the best for P_m. Considering the above analysis, the best combinations of parameter values are $PS = 150$, $P_c = 1$, and $P_m = 0.01$.

The effect of each parameter on the performance of NSGA-II is presented at Table VI for the average total makespan and average total cost. P_m has the most effect as it is ranked first for both cases, while P_c has the lowest effect in the case of the average total makespan and PS has the lowest effect in the case of the average total cost.

B. Detailed Results for fog, cloud, and fog-cloud

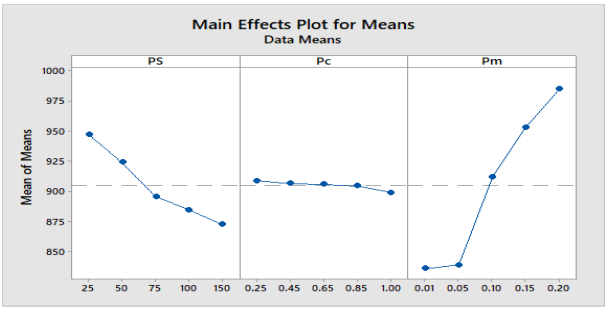
This section discusses the detailed results obtained from both CNGSA-II and DNSGA-II for fog, fog-cloud and cloud systems. After determining the best parameter combination as described in section V-A, the proposed algorithm is used to solve the entire instances for the three systems. Table VII presents the detailed results obtained from solving the 33 instances (11 fog, 11 fog-cloud and 11 cloud) using discrete NSGA-II. As can be concluded from Table VII, the makespan of the fog-cloud system is better while the cost of the the fog system is better. The detailed results obtained from continuous NSGA-II are presented in Table VIII. The makespan obtained for fog is better than fog-cloud and cloud systems for most of the instances, while the cost is always better in the fog system.

The summary of the results obtained from DNSGA-II and CNSGA-II for the three systems are presented at Table IX.In

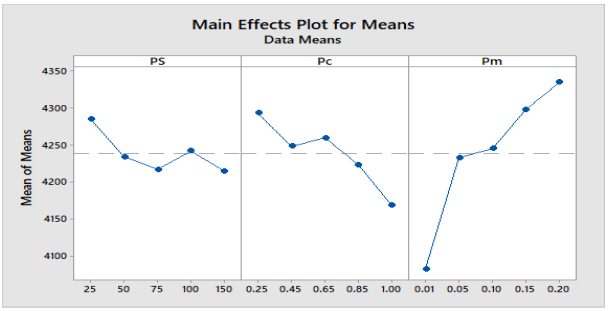
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3032386, IEEE Transactions on Cloud Computing

TABLE V
ORTHOGONAL TABLE AND AVERAGE MAKESPAN AND AVERAGE TOTAL COST VALUES

Experiment Number	Factor			Average Makespan	Average Total Cost
	PS	P_c	P_m		
1	1	1	1	868.51	4281.77
2	1	2	2	923.79	4261.61
3	1	3	3	951.02	4261.13
4	1	4	4	986.33	4316.3
5	1	5	5	1007.2	4303.65
6	2	1	2	856.69	4308.21
7	2	2	3	929.78	4219.28
8	2	3	4	988.84	4357.67
9	2	4	5	984.73	4359.64
10	2	5	1	861.06	3925.39
11	3	1	3	918.17	4274.75
12	3	2	4	909.16	4285.27
13	3	3	5	992.6	4334.27
14	3	4	1	848.18	3992.58
15	3	5	2	810.95	4197.37
16	4	1	4	940.7	4315.36
17	4	2	5	979.11	4389.27
18	4	3	1	811.31	4126.3
19	4	4	2	818.64	4178.38
20	4	5	3	874.15	4200.45
21	5	1	5	959.76	4286.72
22	5	2	1	791.49	4085.27
23	5	3	2	785.11	4218.23
24	5	4	3	885.7	4268.45
25	5	5	4	941	4214.35



(a)



(b)

Fig. 2. Factor level trend of NSGA-II based on (a) average total makespan and (b) average total cost

the fog, based on makespan, DNSGA-II is better than, similar to, and worse than CNSGA-II in 5, 0, 6 instances, respectively. Based on cost, DNSGA-II is superior, equal and inferior to CNGSA-II in 6, 0, 5 instances, respectively. For both fog-cloud and cloud, based on both makespan and cost, DNSGA-II is better than CNGSA-II in the entire test instances.

Considering the Wilcoxon test, DNSGA-II is significantly better than CNGSA-II for both fog-cloud and cloud systems for both cost and makespan, while there is no significant

TABLE VI
RESPONSE TABLE OF MEANS OF BOTH AVERAGE TOTAL MAKESPAN AND AVERAGE TOTAL COST

Criteria	Average Total makespan			Average Total Cost		
Level	PS	P_c	P_m	PS	P_c	P_m
1	947.4	908.8	836.1	4285	4293	4082
2	924.2	906.7	839	4234	4248	4233
3	895.8	905.8	911.8	4217	4260	4245
4	884.8	904.7	953.2	4242	4223	4298
5	872.6	898.9	984.7	4215	4168	4335
Delta	74.8	9.9	148.6	70	125	252
Rank	2	3	1	3	2	1

difference between both algorithms in the case of the fog system as observed in the last column of Table IX. A further analysis is conducted by plotting the performance profiles, which compare DNSGA-II and CNSGA-II in the three systems, as shown in Figure 3. It is clear that DNSGA-II is always better than CNSGA-II as it was able to attain the probability of 1 for all cases first. For more analysis, the Pareto fronts are plotted in Figure 4, which compares the performance of DNSGA-II and CNSGA-II. To sum up, the graphs illustrate that DNSGA-II is better than CNSGA-II.

Based on the above analysis, it can be concluded that, the discrete version of NSGA-II (DNSGA-II) is always better than the continuous version (CNSGA-II). Therefore, we used DNSGA-II to conduct a comparison between fog, fog-cloud and Cloud environments based on both average total cost and average total makespan. As we can see at Table VII, the fog environment has the lowest cost for all instances in comparison with fog-cloud and Cloud systems, while fog-cloud has the lowest makespan for all instances. Overall, if cost matters, our suggestion to users is to use the fog environment, while if the makespan is more important than cost, we recommend the use of fog-cloud environment.

C. Comparison with state-of-the-art-algorithms

This section discusses the performance of the proposed DNSGA-II algorithm compared with other 10 compelling optimisation-based tasking scheduling algorithms for solving single and multiple objectives problems.

1) Comparison of single objective algorithms

The proposed DNSGA-II is designed for multi-objective problems, while considering two conflicting/non-conflicting objectives. However, to compare with several single objective algorithms, two objectives (i.e., makespan and total cost) are transformed into one objective equation using the following equation:

$$Minimize : F \tag{21}$$

where

$$F = 0.5 \times Makespan + 0.5 \times TotalCost \tag{22}$$

In Equation 22, both makespan and total cost are given equal weights (0.5) to transform into a single objective framework, which is a very common practice in the relevant literature [10]. In the fog system, the performance of the proposed algorithm is compared with four algorithms: 1) Time-Cost aware Scheduling (TCaS) [10], 2) Bee Life Algorithm (BLA) [43], 3) Modified Particle Swarm Optimization (MPSO) [44] and 4) a simple Round Robin (RR) [45]. The outputs of the

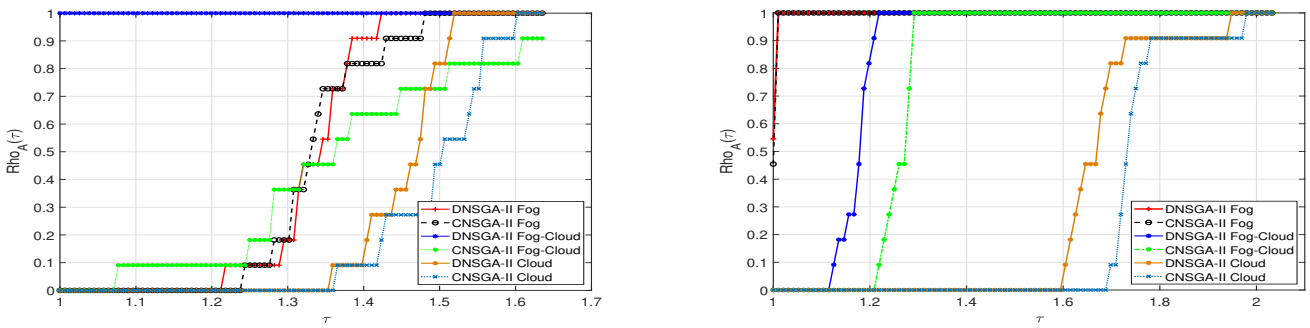
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3032386, IEEE Transactions on Cloud Computing

TABLE VII
SUMMARY OF DETAILED MAKESPAN AND TOTAL COST VALUES OBTAINED FROM DISCRETE NSGA-II

Instances	Fog (10 Nodes)			Fog-cloud (13 Nodes)			Cloud (3 Nodes)		
	Makespan	**Cost**	**Time (s)**	**Makespan**	**Cost**	**Time (s)**	**Makespan**	**Cost**	**Time (s)**
40	114.08	**366.88**	16.24	**82.60**	415.96	18.03	116.29	711.93	**10.70**
80	335.11	**968.35**	16.44	**236.08**	1146.31	18.85	347.84	1628.88	**10.99**
120	455.74	**1428.83**	16.55	**330.88**	1675.31	19.98	498.97	2280.86	**11.59**
160	600.72	**1882.60**	17.45	**455.36**	2224.95	20.05	637.73	3184.72	**12.08**
200	717.62	**2348.81**	18.64	**590.46**	2610.73	21.10	800.62	3893.36	**12.92**
250	918.34	**2983.07**	19.48	**699.80**	3484.21	21.92	1021.05	4845.92	**13.78**
300	1147.54	**3645.65**	20.38	**847.47**	4371.47	22.36	1265.57	5857.88	**14.38**
350	1298.67	**4251.81**	21.14	**959.44**	5126.33	24.03	1455.04	6856.27	**15.17**
400	1382.51	**4536.45**	22.08	**1072.17**	5239.97	24.34	1541.65	7447.39	**15.85**
450	1691.43	**5436.82**	23.71	**1258.32**	6493.59	25.90	1860.82	9351.24	**17.44**
500	1930.72	**6256.01**	25.32	**1470.50**	7333.50	26.84	2172.93	10384.31	**18.43**
Average	962.95	**3100.48**	19.77	**727.55**	3647.48	22.13	1065.32	5131.16	**13.94**

TABLE VIII
SUMMARY OF DETAILED MAKESPAN AND TOTAL COST VALUES OBTAINED FROM CONTINUOUS NSGA-II

Instances	Fog (10 Nodes)			Fog-cloud (13 Nodes)			Cloud (3 Nodes)		
	Makespan	**Cost**	**Time (s)**	**Makespan**	**Cost**	**Time (s)**	**Makespan**	**Cost**	**Time (s)**
40	122.02	**369.06**	23.56	**88.44**	443.97	23.26	117.61	722.92	**15.57**
80	337.18	**974.92**	27.50	**302.10**	1231.30	26.92	355.38	1665.51	**18.03**
120	**454.48**	**1428.43**	28.98	456.33	1755.24	29.93	514.75	2445.38	**21.01**
160	592.76	**1892.51**	32.68	**581.51**	2362.52	33.28	647.24	3344.55	**22.52**
200	**734.30**	**2329.33**	35.06	735.59	2981.99	37.37	804.54	4075.83	**26.55**
250	**932.15**	**2965.32**	39.59	954.27	3773.83	44.58	1042.68	5144.04	**27.76**
300	**1122.49**	**3648.44**	42.99	1223.29	4672.63	44.42	1318.17	6173.32	**32.48**
350	**1285.61**	**4225.41**	46.67	1538.02	5213.55	46.96	1536.37	7300.92	**37.28**
400	**1374.12**	**4563.22**	50.21	1410.77	5822.98	52.87	1600.14	7979.26	**38.62**
450	**1639.38**	**5448.32**	55.24	1898.97	7005.68	55.80	1928.88	9424.87	**44.32**
500	**1972.06**	**6209.10**	58.78	2513.82	7725.56	57.75	2263.39	10637.89	**47.28**
Average	**960.60**	**3095.82**	40.11	1063.92	3908.11	41.19	1102.65	5355.86	**30.13**



(a) (b)

Fig. 3. Performance profile graphs comparing DNSGA-II and CNSGA-II for fog, fog-cloud and Cloud based on (a) makespan; (b) based on cost

TABLE IX
COMPARISON SUMMARY BETWEEN DNSGA-II AND CNSGA-II FOR FOG, FOG-CLOUD AND CLOUD SYSTEMS BASED ON BOTH AVERAGE TOTAL MAKESPAN AND AVERAGE TOTAL COST

	Algorithms	Better	Equal	Worse	Dec.
Makespan	DNSGA-II vs. CNSGA-II (Fog System)	5	0	6	≈
	DNSGA-II vs. CNSGA-II (Fog-cloud System)	11	0	0	+
	DNSGA-II vs. CNSGA-II (Cloud System)	11	0	0	+
Cost	DNSGA-II vs. CNSGA-II (Fog System)	6	0	0	≈
	DNSGA-II vs. CNSGA-II (Fog-cloud System)	11	0	0	+
	DNSGA-II vs. CNSGA-II (Cloud System)	11	0	0	+

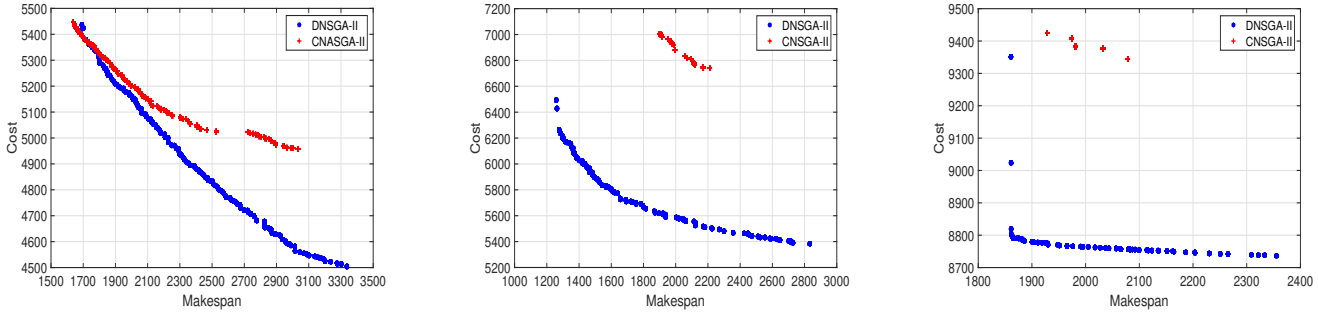
compassion's between the algorithms are listed in Table X. In the fog-cloud system, besides the four algorithms above, other two algorithms: 5) Genetic Algorithm (GA) [46] and 6) Differential Evolution (DE) [47] are used. The results of the comparisons are demonstrated in Table XI. From both tables, the proposed DNSGA-II can obtain the best F values for all instances in both fog and fog-cloud systems. This indicates the superior performance of the proposed algorithm in achieving better total costs and makespan than other comparative single objective-based algorithms.

A further analysis is conducted based on the Wilcoxon rank-test listed in Table XII. It can be noticed that the performance of the proposed DNSGA-II is significantly better than all the rival algorithms in fog and fog-cloud systems based on the resultant F values shown in equation 22.

2) Comparison of multi-objective algorithms

In this section, the performance of the proposed DNSGA-II is compared with four Multi-objectives algorithms: 1) Strength

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3032386, IEEE Transactions on Cloud Computing



(a) (b) (c)

Fig. 4. Pareto front comparing results obtained from DNGSA-II and CNSGA-II for instances with 500 tasks (a) Fog environment; (b) Fog-cloud environment; and (c) Cloud environment

TABLE X
COMPARISONS OF DNSGA-II AGAINST SOME STATE-OF-THE-ART ALGORITHMS BASED ON F VALUES FOR FOG ENVIRONMENT

Instances	*F*				
	DNSGA-II	**TCaS**	**BLA**	**MPSO**	**RR**
40	**243.54**	462.65	469.23	477.83	606.05
80	**650.99**	967.46	978.47	999.39	1272.45
120	**949.25**	1485.54	1511.26	1533.68	1957.28
160	**1233.85**	1998.07	2050.74	2064.67	2579.39
200	**1531.94**	2348.04	2417.43	2422.11	2870.55
250	**1958.24**	3129.95	3253.91	3243.72	4066.93
300	**2394.73**	3653.24	3821.45	3787.64	4622.54
350	**2754.71**	4128.28	4332.06	4271.83	5188.19
400	**2965.68**	4841.57	5119.01	5030.23	6111.52
450	**3539.89**	5540.53	5883.68	5813.80	7183.47
500	**4100.59**	6202.87	6585.18	6494.51	8060.75

Pareto Evolutionary Algorithm 2 (SPEA-II) [48], 2) Multi-objective Particle Swarm Optimization algorithm (MOPSO) [49], 3) Pareto Envelope-based Selection Algorithm II (PESA-II) [50], and 4) ulti-objective Evolutionary Algorithm based on Decomposition (MOEA/D) [51]. These algorithms are used as they showed high performances in solving multi-objective optimization problems.

The parameter settings of these algorithms are taken from their original paper, and the stopping condition (maximum number of iteration) for all of them including the proposed algorithm is set to 5000 generations. Also, they run for 30 times and their best results are recorded in Table XIII.

The makespan and cost obtained from DNSGA-II and the competing algorithms are presented in Table XIII. It is clear that the proposed DNSGA-II is better than others in a term of makespan. Considering the cost objective function, DNSGA-II is better than SPEA-II, MOPSO, PEAS-II and MOEA/D in 8, 9, 9 and 9 instances, respectively. The proposed DNSGA-II algorithm can save average makespan considering all instances by 36.75%, 39.15%, 41.99% and 41.57% compared to SPEA-II, MOPSO, PEAS-II and MOEA/D, respectively. DNSGA-II can save the average cost of all instances by 2.72%, 5.55%, 3.25% and 3.92%, compared to PEA-II, MOPSO, PEAS-II and MOEA/D, respectively.

The Wilcoxon test is also carried out to judge if there is a statistical significant difference between the proposed DNSGA-II and the other algorithms, where the results are presented in Table XIV. From Table XIV, it is clear that the proposed DNSGA-II statistically outperforms the rival algorithms for both makespan and cost. Additionally, the Pareto-fronts obtained from DNSGA-II, SPEA-II, MOPSO, PESA-II and MOEA/D algorithms are plotted for instances with 40 and 450 tasks, depicted in Figure 5. It is obvious that the proposed DNSGA-II algorithm can produce better results (solutions are in the lower areas) than other rival algorithms. Based on these experiments for both single and multi-objective formulations, the proposed DNSGA-II is empirically proven that it is the best one compared with the competing algorithms.

D. Discussions and Implications

This section explains the sensitivity analysis of the proposed DNSGA-II algorithm for fog and fog-cloud systems. One fundamental query regarding task scheduling or latency minimization is to sort out an optimal number of nodes for each system. The optimum number of nodes for different computing systems, a practitioner can allocate resources accordingly based on employing the proposed algorithm. It can be seen from Figure 6, both cost and makespan are drawn for executing different tasks or instances of different nodes in the fog and fog-cloud systems. Since, comparing with two contrasting values is very tricky, we hypothetically give 50% credit to each cost and makespan value, i.e., each cost and makespan values are multiplied by 0.5 and then summed up to calculate normalised value, as shown in that figure. Hence, $normalisedvalue = 0.5 * cost + 0.5 * makespan$. In the case of Figure 6(a), normalised values under the fog system show a sharp decreasing trend with the increasing number of fog nodes up to 10 fog nodes (i.e., 10FN), after which the decrements flatten. Hence, we can claim that for the fog system, if we intend to solve different tasks or instances, having 16 fog nodes should be the optimum decision, considering both makespan and cost values.

Meanwhile, for the fog-cloud environment, this work started considering 3 cloud nodes with different fog nodes. As depicted in Figure 6(b), the normalised value of cost and makespan decreased up to 3 cloud nodes and 16 fog nodes (i.e., 3CN&16FN), and then flattened. To identify the optimum number of cloud nodes, another experiment was carried out with 16 fog nodes and different cloud nodes for the same

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2020.3032386, IEEE Transactions on Cloud Computing

TABLE XI
COMPARISONS OF DNSGA-II AGAINST SOME STATE-OF-THE-ART ALGORITHMS BASED ON F VALUES FOR FOG-CLOUD ENVIRONMENT

Instances	F						
	DNSGA-II	**TCaS**	**BLA**	**MPSO**	**RR**	**GA**	**DE**
40	**249.28**	487.95	498.36	484.43	586.49	597.72	607.50
80	**691.20**	1025.41	1050.62	1020.88	1379.43	1414.88	1421.74
120	**1003.10**	1568.57	1610.04	1565.13	1977.44	1676.20	2069.79
160	**1340.16**	2105.04	2159.88	2113.87	2658.42	2608.00	2652.67
200	**1600.60**	2518.46	2571.06	2503.58	2922.68	2692.65	3044.84
250	**2092.01**	3151.22	3238.47	3133.89	3800.75	3722.81	3875.03
300	**2609.47**	3896.64	4014.17	3890.33	4764.27	4444.26	4797.27
350	**3042.89**	4453.78	4582.14	1167.54	5223.22	4996.00	5351.96
400	**3156.07**	5182.03	5344.60	1373.58	5965.94	5870.29	6391.28
450	**3875.96**	5939.37	6142.45	5925.58	6839.80	6781.50	7201.34
500	**4402.00**	6646.87	6865.79	6624.85	7949.90	7446.65	7644.54

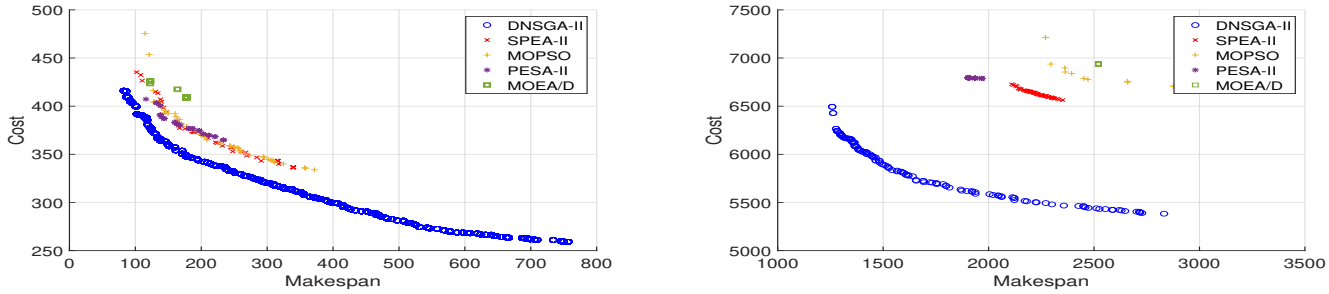


Fig. 5. Comparison of Pareto fronts obtained from DNGSA-II, SPEA-II, MOPSO, PESA-II and MOEA/D for Fog-Cloud system for instances with (a) 40 instances; and (b) 450 instances

TABLE XII
COMPARISON SUMMARY OF DNSGA-II WITH OTHER ALGORITHMS FOR FOG AND FOG-CLOUD SYSTEMS

. Criteria	Algorithms	Better	Equal	Worse	Dec.
Fog environment	DNSGA-II vs. TCaS	11	0	0	+
	DNSGA-II vs. BLA	11	0	0	+
	DNSGA-II vs. MPSO	11	0	0	+
	DNSGA-II vs. RR	11	0	0	+
Fog-cloud environment	DNSGA-II vs. TCaS	11	0	0	+
	DNSGA-II vs. BLA	11	0	0	+
	DNSGA-II vs. MPSO	11	0	0	+
	DNSGA-II vs. RR	11	0	0	+
	DNSGA-II vs. GA	11	0	0	+
	DNSGA-II vs. DE	11	0	0	+

fog-cloud environment. It can be seen from Figure 6(c), the normalised values are increasing with increasing cloud nodes. Hence, from this experiment, it can be claimed that for the fog-cloud environment, having 16 fog nodes and 1 cloud node (i.e., 16FN&1CN) is the optimal decision, considering both makespan and cost.

Another experiment is also carried out to observe performance measures with increasing task numbers. As expected, normalised values of cost and makespan always show a non-linearly increasing trend with increasing task numbers for both fog and fog-cloud environments, as depicted in Figure 7. It is observed in Figure 8, the computational time in seconds also increases with the increasing number of nodes for both environments. CT_fog and CT_cloud_fog represent the computing time in seconds for fog and fog-cloud environments, respectively. It is noted that the computing time of the fog-cloud environment is always higher than the simple fog nodes. This is also obvious, even in the case of fog-cloud nodes, as there are more tasks that need to be allocated in a higher number of nodes.

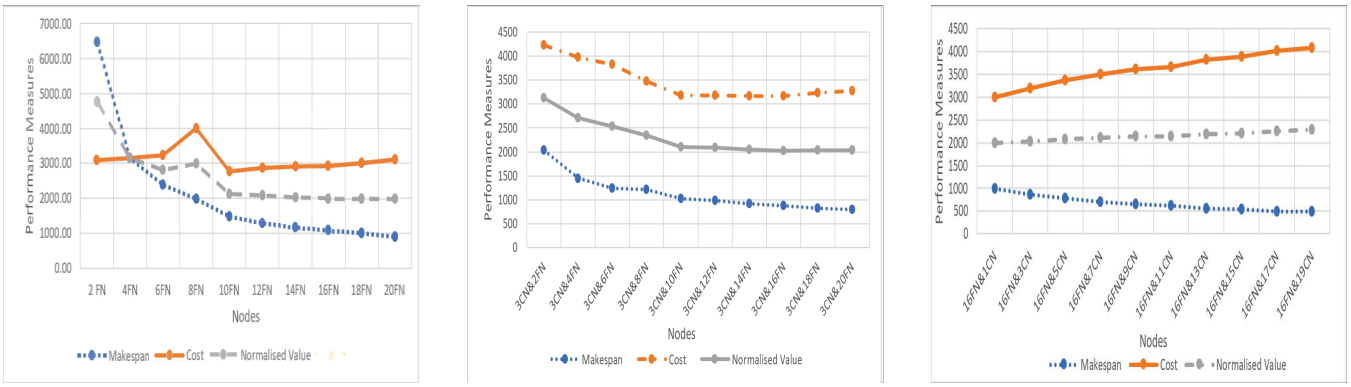
VI. CONCLUSION AND FUTURE WORK

This paper has introduced an enhanced discrete non-dominated sorting genetic algorithm II (DNSGA-II) for solving the problem of task scheduling and resource allocation in fog-cloud environments. We first formulated the communication between fog and cloud tiers as a multi-objective optimization problem that allows auto-tuning of the parameters of the proposed DNSGA-II model in a dynamic environment. Then, we have implemented the proposed model in a simulated environment to determine its capability to allocate tasks to proper computing resources either in the fog or the cloud. The model has proven its high performance in distributing the tasks to appropriate resources in fog-cloud systems. Several statistical analyses and experiments were carried out to examine the model's evaluation. The model was also compared with a continuous NSGA-II model and other four peer mechanisms to determine how tasks and their needed resources could properly be allocated either in the fog or the cloud layer. The experimental results showed the high performance of the proposed model in terms of lower makespans and costs compared with others. This could give an indication to utilize the model in distributing batch tasks with large-scale data in fog-cloud environments.

In the future, we will extend this work to address the challenge of allocating resources in cyber-physical systems and also allow auto-configuration of service orchestration at the edge of a network.

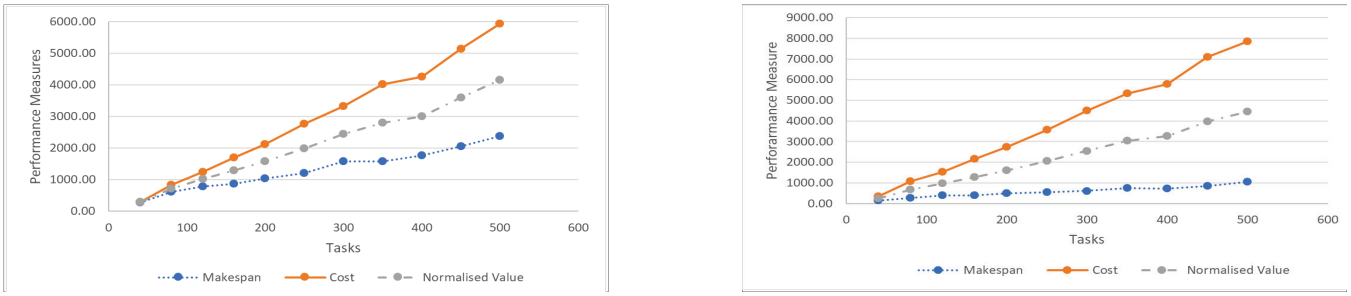
TABLE XIII
COMPARISONS OF DNSGA-II WITH SPEA-II, MOPSP, PESA-II AND MOEA/D BASED ON BOTH MAKESPAN AND TOTAL COST FOR FOG-CLOUD SYSTEM

Instances	Makespan					Cost				
	DNSGA-II	SPEA-II	MOPSO	PESA-II	MOEA/D	DNSGA-II	SPEA-II	MOPSO	PESA-II	MOEA/D
40	**82.60**	116.82	111.88	123.90	156.08	415.96	481.80	**407.06**	438.35	414.79
80	**236.08**	383.43	381.34	314.12	439.36	1146.31	1258.15	1232.43	1063.97	**997.73**
120	**330.88**	503.98	423.96	490.41	608.35	1675.31	**1649.36**	1791.79	1765.31	1679.85
160	**455.36**	721.24	629.03	762.35	712.94	**2224.95**	2456.32	2808.84	2324.57	2296.11
200	**590.46**	1040.92	905.53	838.43	823.73	**2610.73**	3044.48	2649.51	2625.04	2687.78
250	**699.80**	1042.74	1058.35	1252.49	1128.23	**3484.21**	3488.35	3610.48	3631.06	3780.91
300	**847.47**	1358.16	1647.12	1444.33	1431.35	4371.47	**4363.22**	4671.82	4430.15	4470.30
350	**959.44**	1659.61	1671.66	1680.00	1606.41	5126.33	**4908.52**	4935.09	5035.62	5172.39
400	**1072.17**	1695.57	1703.28	1838.97	2002.29	**5239.97**	5678.74	5923.38	5619.17	5833.92
450	**1258.32**	1783.53	2096.77	2350.23	2359.27	**6493.59**	6503.91	6591.60	6782.36	6978.20
500	**1470.50**	2346.92	2524.08	2702.78	2428.57	**7333.50**	7409.96	7859.39	7755.05	7447.02
Avg.	**727.55**	1150.26	1195.73	1254.36	1245.14	**3647.48**	3749.34	3861.94	3770.06	3796.27



(a) (b) (c)

Fig. 6. Performance measures based on number of nodes (a) fog environment; (b) fog-cloud environment (3 cloud nodes and different fog nodes); and (c) fog-cloud environment (16 fog nodes and different cloud nodes)



(a) (b)

Fig. 7. Performance measures based on number of tasks (a) fog environment; (b) fog-cloud environment

TABLE XIV
COMPARISON SUMMARY BETWEEN DNSGA-II, SPEA-II, MOPSO, PESA-II AND MOEA/D FOR FOG-CLOUD SYSTEM BASED ON BOTH MAKESPAN AND COST.

Criteria	Algorithms	Better	Equal	Worse	Dec.
Makespan	DNSGA-II vs. SPEA-II	11	0	0	+
	DNSGA-II vs. MOPSO	11	0	0	+
	DNSGA-II vs. PESA-II	11	0	0	+
	DNSGA-II vs. MOEA/D	11	0	0	+
Cost	DNSGA-II vs. SPEA-II	8	0	3	+
	DNSGA-II vs. MOPSO	9	0	2	+
	DNSGA-II vs. PESA-II	9	0	2	+
	DNSGA-II vs. MOEA/D	9	0	2	+

REFERENCES

[1] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *2016 18th Asia-Pacific network operations and management symposium (APNOMS)*. IEEE, 2016, pp. 1–4.

[2] 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.

[3] S. Basu, M. Karuppiah, K. Selvakumar, K.-C. Li, S. H. Islam, M. M. Hassan, and M. Z. A. Bhuiyan, "An intelligent/cognitive model of task scheduling for iot applications in cloud computing environment," *Future Generation Computer Systems*, vol. 88, pp. 254–261, 2018.

[4] J. Hartmanis, "Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson)," *Siam Review*, vol. 24, no. 1, p. 90, 1982.

[5] D. Chahal, B. Mathew, and M. Nambiar, "Simulation based job scheduling optimization for batch workloads," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*,

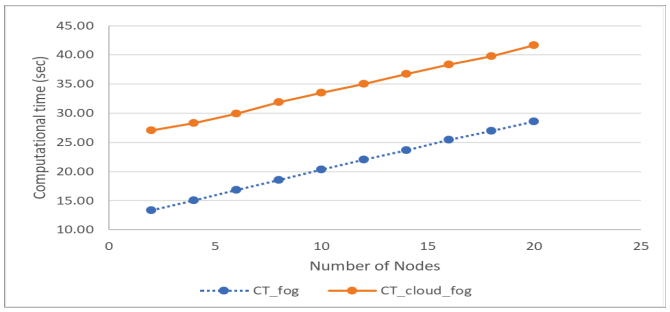


Fig. 8. Computational time for different computing environments

2019, pp. 313–320.

[6] Z. Liu, Y. Yang, M.-T. Zhou, and Z. Li, "A unified cross-entropy based task scheduling algorithm for heterogeneous fog networks," in *Proceedings of the 1st ACM International Workshop on Smart Cities and Fog Computing*, 2018, pp. 1–6.

[7] J. Brownlee, "Clever algorithms: Nature-inspired programming recipes," 2011.

[8] M. Mehmood, N. Javaid, J. Akram, S. H. Abbasi, A. Rahman, and F. Saeed, "Efficient resource distribution in cloud and fog computing," in *International Conference on Network-Based Information Systems*. Springer, 2018, pp. 209–221.

[9] N. K. Madavan, "Multiobjective optimization using a pareto differential evolution approach," in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2. IEEE, 2002, pp. 1145–1150.

[10] B. M. Nguyen, H. Thi Thanh Binh, B. Do Son *et al.*, "Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment," *Applied Sciences*, vol. 9, no. 9, p. 1730, 2019.

[11] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018, pp. 103–130.

[12] R. Salimi, N. Bazrkar, and M. Nemati, "Task scheduling for computational grids using nsga ii with fuzzy variance based crossover," *Advances in Computing*, vol. 3, no. 2, pp. 22–29, 2013.

[13] G. Luo, X. Wen, H. Li, W. Ming, and G. Xie, "An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 91, no. 9-12, pp. 3145–3158, 2017.

[14] T. Choudhari, M. Moh, and T.-S. Moh, "Prioritized task scheduling in fog computing," in *Proceedings of the ACMSE 2018 Conference*. ACM, 2018, p. 22.

[15] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Transactions on Cloud Computing*, 2019.

[16] L. Liu, D. Qi, N. Zhou, and Y. Wu, "A task scheduling algorithm based on classification mining in fog computing environment," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[17] C. Tang, X. Wei, S. Xiao, W. Chen, W. Fang, W. Zhang, and M. Hao, "A mobile cloud based scheduling strategy for industrial internet of things," *IEEE Access*, vol. 6, pp. 7262–7275, 2018.

[18] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A hyper-heuristic scheduling algorithm for cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 236–250, 2014.

[19] R. Z. Naeem, S. Bashir, M. F. Amjad, H. Abbas, and H. Afzal, "Fog computing in internet of things: Practical applications and future directions," *Peer-to-Peer Networking and Applications*, vol. 12, no. 5, pp. 1236–1262, 2019.

[20] X.-Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E.-N. Huh, "A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing," *International Journal of Distributed Sensor Networks*, vol. 13, no. 11, p. 1550147717742073, 2017.

[21] 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 Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[22] 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.

[23] H. R. Boveiri, R. Khayami, M. Elhoseny, and M. Gunasekaran, "An efficient swarm-intelligence approach for task scheduling in cloud-based internet of things applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 9, pp. 3469–3479, 2019.

[24] R. Jena, "Multi objective task scheduling in cloud environment using nested pso framework," *Procedia Computer Science*, vol. 57, pp. 1219–1227, 2015.

[25] A. S. Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wireless Personal Communications*, vol. 107, no. 4, pp. 1835–1848, 2019.

[26] W. Wu, H. R. Maier, and A. R. Simpson, "Single-objective versus multiobjective optimization of water distribution systems accounting for greenhouse gas emissions by carbon pricing," *Journal of Water Resources Planning and Management*, vol. 136, no. 5, pp. 555–565, 2010.

[27] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved nsga-ii," *Wireless Personal Communications*, vol. 102, no. 2, pp. 1369–1385, 2018.

[28] Y. Chen, J. Huang, C. Lin, and X. Shen, "Multi-objective service composition with qos dependencies," *IEEE Transactions on Cloud Computing*, 2016.

[29] 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.

[30] R. Gupta and S. J. Nanda, "A binary nsga-iii for unsupervised band selection in hyper-spectral satellite images," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 522–529.

[31] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[32] B. B. Khoo, B. Veeravalli, T. Hung, and C. S. See, "A multi-dimensional scheduling scheme in a grid computing environment," *Journal of Parallel and Distributed Computing*, vol. 67, no. 6, pp. 659–673, 2007.

[33] L. Zhihuan, L. Yinhong, and D. Xianzhong, "Non-dominated sorting genetic algorithm-ii for robust multi-objective optimal reactive power dispatch," *IET generation, transmission & distribution*, vol. 4, no. 9, pp. 1000–1008, 2010.

[34] B. L. Miller, D. E. Goldberg *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[35] A. Jaszkiewicz and J. Branke, "Interactive multiobjective evolutionary algorithms," in *Multiobjective optimization*. Springer, 2008, pp. 179–193.

[36] V. Meruane and W. Heylen, "An hybrid real genetic algorithm to detect structural damage using modal properties," *Mechanical Systems and Signal Processing*, vol. 25, no. 5, pp. 1559–1573, 2011.

[37] I. Korejo, S. Yang, and C. Li, "A directed mutation operator for real coded genetic algorithms," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2010, pp. 491–500.

[38] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1062–1087, 2008.

[39] B. Tang, Z. Chen, G. Hefferman, S. Pei, T. Wei, H. He, and Q. Yang, "Incorporating intelligence in fog computing for big data analysis in smart cities," *IEEE Transactions on Industrial informatics*, vol. 13, no. 5, pp. 2140–2150, 2017.

[40] "fog-cloud task scheduling dataset," November 2019. [Online]. Available: https://research.unsw.edu.au/projects/cross-disciplinary-optimization-under-capability-context.

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

[42] H. J. Barbosa, H. S. Bernardino, and A. M. Barreto, "Using performance profiles for the analysis and design of benchmark experiments," in *Advances in Metaheuristics*. Springer, 2013, pp. 21–36.

[43] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems*, vol. 12, no. 4, pp. 373–397, 2018.

[44] S. Abdi, S. A. Motamedi, and S. Sharifian, "Task scheduling using modified pso algorithm in cloud computing environment," in *International conference on machine learning, electrical and mechanical engineering*, 2014, pp. 8–9.

[45] I. S. Rajput and D. Gupta, "A priority based round robin cpu scheduling algorithm for real time systems," *International Journal of Innovations in Engineering and Technology*, vol. 1, no. 3, pp. 1–11, 2012.

[46] M. D. Vose, *The simple genetic algorithm: foundations and theory*. MIT press, 1999.

[47] I. M. Ali, D. Essam, and K. Kasmarik, "A novel differential evolution mapping technique for generic combinatorial optimization problems," *Applied Soft Computing*, vol. 80, pp. 297–309, 2019.

[48] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[49] G. T. Pulido and C. A. C. Coello, "Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer," in *Genetic and Evolutionary Computation Conference*. Springer, 2004, pp. 225–237.

[50] D. W. Corne, J. D. Knowles, and M. J. Oates, "The pareto envelope-based selection algorithm for multiobjective optimization," in *International conference on parallel problem solving from nature*. Springer, 2000, pp. 839–848.

[51] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.