Model and Algorithms for the Planning of Fog Computing Networks

Decheng Zhang, Faisal Haider, Marc St-Hilaire, *Senior Member, IEEE*, and Christian Makaya, *Member, IEEE*

***Abstract*—Fog computing has risen as a promising technology for augmenting the computational and storage capability of the end devices and edge networks. The urging issues in this networking paradigm are fog nodes planning, resources allocation, and offloading strategies. This paper aims to formulate a mathematical model which jointly tackles these issues. The goal of the model is to optimize the tradeoff (Pareto front) between the capital expenditure and the network delay. To solve this multiobjective optimization problem and obtain benchmark values, we first use the weighted sum method and two existing evolutionary algorithms (EAs), nondominated sorting genetic algorithm II and speed-constrained multiobjective particle swarm optimization. Then, inspired by those EAs, this paper proposes a new EAs, named particle swarm optimized nondominated sorting genetic algorithm, which combines the convergence and searching efficiency of the existing EAs. The effectiveness of the proposed algorithm is evaluated by the hypervolume and inverted generational distance indicators. The performance evaluation results show that the proposed model and algorithms can help the network planners in the deployment of fog networks to complement their existing computation and storage infrastructure.**

***Index Terms*—Computation offloading, evolutionary algorithm (EA), fog computing, heuristic, modular facility location, multiobjective, network planning.**

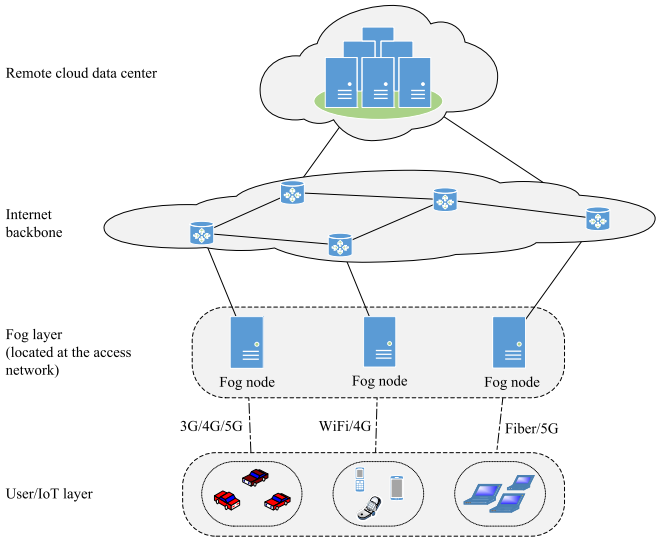


Fig. 1. Typical fog network architecture.

I. Introduction

THE conventional cloud computing paradigm has been applied as a solution for delivering a diverse range of services. However, the long network distance between edge devices and remote data centers hinders the service quality, resulting in high latency (delay), high bandwidth consumption, and unreliable connectivity [1]. Moreover, the emerging and increasing deployment of Internet of Things (IoT) applications bring new challenges and affect the approach used to connect the edge networks with the remote data centers. For instance, the IoT applications usually require mobility support, geo-distribution, location-awareness, near real-time processing, and low latency [2]. The traditional cloud system can hardly provide these requirements and still has unsolved problems in service management, connectivity, and security challenges [3].

Manuscript received November 7, 2018; revised December 14, 2018; accepted January 7, 2019. Date of publication January 14, 2019; date of current version May 8, 2019. *(Corresponding author: Marc St-Hilaire.)*

D. Zhang, F. Haider, and M. St-Hilaire are with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON K1S 5B6, Canada (e-mail: marc_st_hilaire@carleton.ca).

C. Makaya is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA.

Digital Object Identifier 10.1109/JIOT.2019.2892940

To address the aforementioned challenges, Cisco proposed the concept of fog computing. The idea of fog computing is to bring computation, communication, and storage closer to the end-users or edge networks [3]. In this paradigm, applications and workloads are offloaded to local micro-data centers (called fog nodes). Since services are hosted in the vicinity of the end-users, the number of hops is reduced thus minimizing the response time for the applications. In addition, fog computing can alleviate the traffic congestion in the Internet backbone since the huge amount of traffic generated by the end devices can be locally handled or cached by the fog nodes. Typically, the fog architecture can be described as shown in Fig. 1. The fog architecture is deployed as a large number of fog nodes (i.e., micro-data centers), which are often distributed over a geographical region of interest. The fog nodes connect to the access networks and remote clouds to provide storage and computing services without the intervention of third-parties [4].

The integration of IoT with cloud computing is considered as one of the primary uses of the fog computing. For instance, fog computing facilitates the deployment of time-sensitive IoT applications, such as augmented reality [5], smart city [6], online gaming [7], worker safety in industrial plant, etc. [8]–[11]. Nevertheless, the integration poses potential

challenges for the computation and communication infrastructure. One of the major challenges is the fog network design. A proper design of the fog network is crucial since it allows the providers to minimize their costs and users can enjoy a good quality of experience. Considering the decentralized nature of the fog network, combined with a large number of edge devices, obtaining the optimal network design is challenging. The following factors must be taken into account.

1) Fog facilities need to be highly decentralized and close to the edge devices and networks.
2) The fog architecture needs to take into account heterogeneous hardware requirements.
3) User demands need to be distributed across multiple facilities.
4) The remote cloud center needs to be able to connect and cooperate with the fog facilities.

To address this network planning problem, a robust mathematical model is necessary. Moreover, an integrated framework jointly tackling these issues is crucial both for the service providers to minimize their costs and for users to enjoy a good quality of experience.

Typically, single-objective optimization is employed in network planning. The designing process is essentially an allocation problem which maximizes the network utility function over a fixed capital expenditure (CAPEX) constraint. However, in network planning, it is possible to greatly improve the network performance whilst incurring a very small increase in CAPEX. Such tradeoff solutions can be easily missed using the single objective method. To obtain an optimal solution, network designers must use a form of trial and error to try different budget options. The inability to derive an analytical perspective of the problem requires the network designers to execute the single-objective optimization multiple times, which significantly increases the computational overhead and might ignore the relation or constraints between requirements.

In contrast, multiobjective optimization (MOO) searches for the optimal set of tradeoff solutions while simultaneously taking all the objectives into account. Hence, it can easily catch the aforementioned tradeoff solutions. Moreover, since the MOO techniques attempt to find multiple nondominated and isolated solutions within a single simulation (by emphasizing multiple solutions toward the Pareto-front), it speeds up the optimization process. Therefore, MOO is used in this paper to model the fog network planning problem.

It is worth noting that a recent comprehensive survey on the state-of-the-art of fog computing research acknowledged that the planning and design of fog networks deployment received very little attention in the literature so far [12]. Besides, to the best of our knowledge, the joint design of computation offloading, resource allocation, and facility provisioning has not been addressed in the previous work. The contributions of this paper can be summarized as follows.

1) We formulate an MOO model to deal with the planning and design of fog networks. More precisely, the model simultaneously considers the offloading decision, the facility location, and the facility provisioning.
2) The model formulation and optimal results obtained with the mixed-integer problem (MIP) toolbox can be used as benchmark for the development of heuristic algorithms.
3) We apply the weighted sum method and two existing evolutionary algorithms (EAs), nondominated sorting genetic algorithm II (NSGA-II) and speed-constrained multiobjective particle swarm optimization (SMPSO), to solve the multiobjective fog planning problem (FPP).
4) We propose a new EA, namely particle swarm optimized nondominated sorting genetic algorithm (PSONSGA), to solve the problem in an efficient and practical way. Simulation results show the effectiveness of the proposed algorithm with different benchmarks and comparison tests.

The remainder of this paper is organized as follows. In Section II, the FPP formulation is presented. In Section III, the NP-hardness of the fog planning (FP) model is analyzed. Section IV presents a comprehensive explanation of the multiobjective algorithms and optimization procedures. Numerical results are presented in Section V, including the evaluation and comparison in terms of the solution quality and computation time. In Section VI, we overview some related work in the area of fog computing and distributed cloud computing planning. Finally, conclusions are drawn in Section VII.

II. Problem Formulation

To help understand the model, we first introduce the following basic concepts.

Edge Cluster: The notion of edge cluster represents an aggregation of user requests. Typically, several users are using the cloud at the same time from a common geographical region sharing a unique IP address prefix. Instead of modeling them individually, we aggregate them together as an "edge cluster." In other words, the planning is not based on each individual user request, but based on an abstraction of grouped requests. This is also similar to the concept of regional request rate. This methodology is commonly applied in the network planning research area [13], [14]. In this paper, the terms "user," "edge device," and edge cluster have the same meaning. It is also important to note that the planning is made from traffic forecast that could represent the maximum amount of expected traffic (plus an extra margin for safety). The dynamic of the day-to-day management can be done via other load balancing and resource allocation schemes which is beyond this topic and can be exploited in future research.

Resource Demand: In this paper, two types of resources are considered at the fog nodes: virtual central processing unit (vCPU) cores and memory. However, other resources can be included such as, storage and graphical processing unit (GPU) units. This can be achieved by increasing the dimension of the fog profile. For any single period of time, each user-cluster generates a request consisting of vCPU, memory, and bandwidth demands. If a request can be routed to a fog node in such a way that the required vCPU, memory, and link bandwidth can be satisfied, then the request is offloaded to this fog node; otherwise, the request is sent to the cloud.

Fog Type: The fog type is an abstraction of the real-world machine servers. The fog types are associated with different computation resources. In real-world planning, the number of fog types and fog profiles can be changed to adequate types or amounts.

Link Type: Similarly, different link types between the fog nodes and the cloud are considered. Each link type is associated with a bandwidth capacity. These links carry the traffic flow between the fog nodes and the cloud for back-end services, such as data synchronizations and application management.

A. Assumptions

This section describes the assumptions used in this paper. Note that the mathematical model has been derived from the proposed hierarchical architecture by the OpenFog Consortium, which is a collaborative work between industry and academia. Given a set of edge user's requests, the model simultaneously determines the optimal location, the number, and the capacity of the fog nodes as well as the connection between the fog nodes and the cloud. Respecting the Pareto optimality law, the multiobjective solutions guarantee that no better network performance can be achieved without increasing the CAPEX.

To formulate the FP model, we assume the following information is known.

1) The fog network design assumes that we do a green field design. In other words, there is no existing fog infrastructure in place.
2) The locations of the edge devices and the candidate locations of all the fog nodes (i.e., x and y coordinates) are known. In the case of each edge device, the generated traffic is known.
3) The characteristics (e.g., memory and vCPU) of the different types of fog nodes that may be installed in the network.
4) The bandwidth availability and the cost of each link type. The links are installed between the fog nodes and the cloud.
5) The cloud computing has unlimited resources (e.g., memory and vCPU). We also assume that the cloud is located in a remote location and if a user cannot be served by the fog, it will be routed to the cloud.
6) The fog nodes send a fix ratio (τ) of their total traffic to the cloud data center. The value of τ is a tunable parameter. Different applications will have a different value of τ which captures the amount of traffic that needs to be sent from the fog nodes to the cloud. The traffic includes for example the load for database synchronization, data uploading, service management, etc.

B. Notation

The following notation is defined-based upon the information mentioned above.

1) *Sets:*
   a) $I = \{1, \ldots, i, \ldots, m\}$, set of potential locations to install the fog nodes. Location m represents the remote cloud data center.
      i) c_i^{Rent}, the renting cost for each potential location $i \in I$.
   b) J, set of edge device clusters that must be served by the fog nodes or the cloud. Each edge cluster has an aggregated memory, vCPU and traffic demands.
      i) η_j, the total number of vCPU required by an edge cluster $j \in J$.
      ii) ζ_j, the total amount of memory required by an edge cluster $j \in J$.
      iii) T_j, the total traffic generated by an edge cluster $j \in J$.
      iv) κ_j, the link speed of an edge cluster $j \in J$.
   c) K, set of fog types (or capacity level) that can be installed at different locations. Different fog types $k \in K$ have different specifications.
      i) α^k, the total number of vCPU available for a fog of type $k \in K$.
      ii) λ^k, the total amount of memory available for a fog of type $k \in K$.
      iii) c_k^{Fog}, the cost for a fog of type $k \in K$.
   d) L, set of link types that can be installed at different locations to maintain connections to cloud data centers. Different link types $l \in L$ have different bandwidth capacities.
      i) β^l, the egress bandwidth upper limit for link type $l \in L$.
      ii) c_l^{Link}, the cost ($/meter) for a link of type $l \in L$.
2) *Functions:*
   a) $d_{ab} = \text{Distance}(a, b)$. The Euclidean distance between points a and b. The values of points a and b are the x, y coordinates.
   b) γ, the processing delay. Each router or switch in the data path adds a finite amount of delay as the packet is received, processed, and then forwarded. The processing delay depends on the number of hops between the user and the fog/cloud. It is computed as follows:

$$\text{Processing Delay} : \gamma = r \cdot h \tag{1}$$

   where r is the mean processing delay for each hop (switch or router) and h represents the hop count.
   c) ψ, the transmission delay. The time taken for a process to send the information to the transmission medium (fiber or wire). The transmission delay depends on the link speed and the packet size and can be calculated as follows:

$$\text{Transmission Delay} : \psi = \sigma/\kappa \tag{2}$$

   where σ is the packet size (in Bytes) and κ represents the link speed (in Bytes/sec).
   d) μ, the propagation delay. It equals to the time taken for the signal to propagate from the source to the destination. The propagation delay depends on the medium used. The propagation speed of wireless communication is the speed of light (t). For copper wires, the speed varies from $0.59t$ to $0.79t$ [15]. In this paper, we use $0.59t$ for the speed of copper

wire. The propagation delay is calculated as

$$\text{Propagation Delay} : \mu = \frac{d_{ab}}{(0.59 \cdot t)} \tag{3}$$

where d_{ab} is the Euclidean distance (in Km) between edge cluster a and fog b.

e) $D(d_{ab})$. The function $D(x)$ is the network usage descriptor. $D(x)$ represents the network usage between each user and the fog facilities, which could represent the network latency experienced by users or the traffic sent to the cloud. This function is transparent to the optimization algorithm. In real life network planning, latency is arguably the most important performance metric. A small increase in the latency can cause substantial service level degradation [16], [17]. Therefore, in our experiment, the network usage function $D(x)$ is modeled as the end-to-end delay.

3) *Decision Variables:*
   a) x_{ij}, a 0-1 variable such that $x_{ij} = 1$ if and only if the edge device cluster $j \in J$ is connected to a fog node installed at location $i \in I$.
   b) y_{ik}, a 0-1 variable such that $y_{ik} = 1$ if and only if a fog of type $k \in K$ is installed at location $i \in I$.
   c) z_{il}, a 0-1 variable such that $z_{il} = 1$ if and only if a link of type $l \in L$ is installed at location $i \in I$ to connect to the cloud.

C. Mathematical Model

Based on the notation presented in Section II-B, we can now formulate the FPP, as follows:

Minimize Cost

$$\text{Minimize}\left[\sum_{i=1}^{m}\sum_{k\in K} y_{ik}c_i^{\text{Rent}} + \sum_{i=1}^{m}\sum_{k\in K} y_{ik}c_k^{\text{Fog}} + \sum_{i=1}^{m}\sum_{l\in L} z_{il}c_l^{\text{Link}}d_{i,\text{cloud}}\right]. \tag{4}$$

Minimize Delay

$$\text{Minimize}\left[\sum_{i=1}^{m}\sum_{j\in J} D\big(d_{ij}\big)x_{ij}\right]. \tag{5}$$

As shown above, to achieve the maximum network performance and cost efficiency, the model simultaneously minimizes the total network delay and the total CAPEX required to deploy the fog network. It is important to note that the queueing delay is not included in (5) since we are solving static scenarios.

Both (4) and (5) are subject to following constraints:

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in J. \tag{6}$$

Equation (6) represents the single source constraints, that ensures that each user connects to exactly one fog node or to the cloud

$$\sum_{k\in K} y_{ik} \le 1 \quad \forall i \in I. \tag{7}$$

Equation (7) reflects the fog uniqueness constraints. They enforce that at most one fog node is installed at a given location. In practice, we can install multiple servers in each location, and each server can have different hardware configurations (e.g., memory sticks, CPU, hard disk drive, GPU, etc.). To reduce the complexity, we generalize different server types and hardware combinations to a fix number of fog types. Under this assumption, each potential location can select an appropriate fog type to accommodate the workload demands. In other words, we cannot install two or more fog nodes at the same location. If the left side of (7) equals to zero, it means that the corresponding location is not selected (i.e., no fog facility will be installed at this location)

$$\sum_{l\in L} z_{il} \le 1 \quad \forall i \in I. \tag{8}$$

Similar to (7), (8) represents the link uniqueness constraints. They ensure that at most one link type will be installed at each location. If (7) equals to zero, the corresponding location is not used and therefore, no link will be installed at this location

$$x_{ij} - \sum_{k\in K} y_{ik} \le 0 \quad \forall i \in I, \forall j \in J. \tag{9}$$

Equation (9) corresponds to the openness constraints. They ensure that users can only connect to a fog that is opened/used

$$\sum_{l\in L} z_{il} \le \sum_{k\in K} y_{ik} \quad \forall i \in I. \tag{10}$$

Equation (10) makes sure that each installed fog node at location i will be connected to the cloud

$$\sum_{j\in J} \eta_j x_{ij} \le \sum_{k\in K} y_{ik}\alpha^k \quad \forall i \in I \tag{11}$$

$$\sum_{j\in J} \zeta_j x_{ij} \le \sum_{k\in K} y_{ik}\lambda^k \quad \forall i \in I. \tag{12}$$

Equations (11) and (12) are the capacity constraints for vCPU and memory at the node level. They ensure that the total resource demand does not exceed each fog node's hardware capacity

$$\sum_{j\in J} x_{ij}T_j \cdot \tau \le \sum_{l\in L} z_{il}\beta^l \quad \forall i \in I. \tag{13}$$

Equation (13) expresses the link capacity constraints. They state that the total bandwidth from fog site $i \in I$ to the cloud cannot exceed the egress link bandwidth upper bound

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in J \tag{14}$$

$$y_{ik} \in \{0, 1\} \quad \forall i \in I, \forall k \in K \tag{15}$$

$$z_{ik} \in \{0, 1\} \quad \forall i \in I, \forall k \in K. \tag{16}$$

Finally, (14)–(16) define the decision variables as binary.

III. NP-Hardness

This section establishes the NP-hardness of the FPP defined in Section II-C. The FPP has two objective functions: 1) CAPEX and 2) network delay. The decision variables consist of the fog placement, fog types, and offloading decisions.

Essentially, FPP is a multiobjective combinatorial problem and is extremely difficult even after relaxing certain constraints. For instance, let us assume that the number of open fog nodes is predetermined (i.e., we know the number and the location of the fog nodes) and that the capacity constraints are relaxed (i.e., fog nodes have infinite capacity) at each location. In this case, the FPP can be reduced to the well-known K-median clustering problem. As proved by Megiddo and Supowit [18], this is an NP-hard problem.

From a different perspective, suppose that CAPEX and network delay can be combined into a single utility function. The objective function becomes

$$\text{Minimize}: \ \alpha \cdot \sum^{FNs} \text{fog capital expenditure} + \beta \cdot \sum^{Users} \text{network delay} \tag{17}$$

where α and β are normalizing constants. Problems of this form are referred to as the capacitated facility location problem (CFLP). It has been proved that exact solution of CFLP is NP-hard [18]. In fact, Fowler *et al.* [19] have proved that even an approximation of CFLP is NP-hard.

As mentioned above, the single objective models of this problem are NP-hard and computationally complicated. Without an efficient approach, the multiobjective model, which combines the optimization of two objectives into a single optimization process, can be even more challenging.

IV. Algorithms for FPP

A. Exact Algorithm for FPP

To solve the multiobjective problem, the most obvious technique is to combine multiple objectives into a single objective. This can be achieved by using the weighted sum method.

1) Weighted Sum Method: The weighted sum approach combines multiobjective functions into a single objective function. In this combination, different objectives are given weight values between 0 and 1. The multiobjective problem can be combined by using the following equations:

$$\min \text{ or } \max \left\{ \sum_{j=1}^{Q} \lambda_j z^j(x) : x \in X \right\} \tag{18}$$

$$0 \le \lambda_j \le 1 \tag{19}$$

$$\sum_{j=1}^{Q} \lambda_j = 1 \tag{20}$$

where λ_j is the weight parameter assigned to each objective in order to capture its relative importance. Each objective function is normalized before computing the resultant objective function.

B. Approximate Algorithms for FPP

In this paper, the evolutionary MOO (EMO) is applied to solve the FPP. EMO is a type of meta-heuristic that mimics the principles of natural selection and survival of the fittest. In EMO, multiple Pareto-optimal solutions are found in a single simulation [20]. The experiment results (discussed later in Section V) show that neither NSGA-II nor SMPSO is capable of producing results exhibiting convergence to the true optimum and a good coverage of the solution space. Therefore, we propose a new EMO algorithm (named PSONSGA). The experiments in the next section show that the proposed PSONSGA algorithm overcomes the drawbacks of NSGA-II and SMPSO.

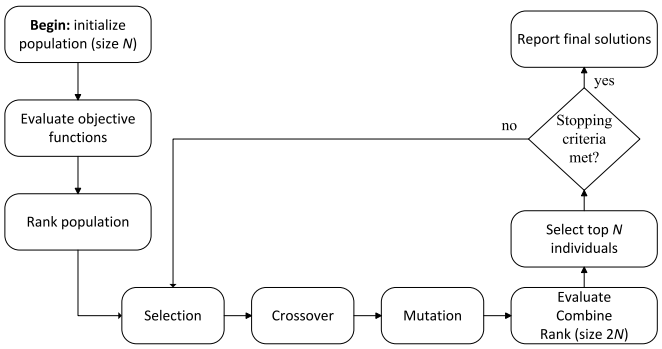


Fig. 2. Steps of the NSGA-II procedure.

1) Nondominated Sorting Genetic Algorithm II: The nondominated sorting genetic algorithm (NSGA) proposed in [21] is one of the most effective multiobjective EAs [22], [23]. With a ranking procedure emphasizing for solutions convergence, NSGA can be expected to find optimal or near-optimal solutions. NSGA-II is a nontrivial revision of NSGA. What differentiates NSGA-II from the previous version of NSGA is its fast elitist ranking procedure. Using this ranking procedure, NSGA-II always preserves the best (higher rank in the nondominated rank) solutions inside the latest population.

The steps to solve the FPP using the NSGA-II procedure are illustrated in Fig. 2. First, a solution set is initialized and evaluated against the objective function [see (4) and (5)]. The solution set is then ranked based on this evaluation results. Second, the promising solutions in higher ranks are selected and different operations (i.e., crossover and mutation) are performed to generate the offspring solutions. Third, all the parent solutions and offspring solutions are combined and ranked again. Higher ranked solutions are stored and carried into the next round of the same procedure. The algorithm is executed repeatedly until the terminating condition is satisfied. When the terminating condition is met, the procedure outputs the best Pareto front solutions found so far.

Based on our preliminary results, we noticed that NSGA-II performs efficiently regarding the convergence to the Pareto front. However, its solution points are unevenly distributed in the search space. Even if we change the mutation index or the crossover index settings, no obvious improvement can be perceived. The researchers in [24] have observed the same disadvantage in continuous NSGA-II applications. This disadvantage stems from the NSGA-II's sorting process. The concentrated effect in the nondominated sorting harms the diversity of the NSGA-II solutions. The researchers in [24] proposed an elitism strategy to overcome this shortcoming. However, the time-complexity involved in evaluating the

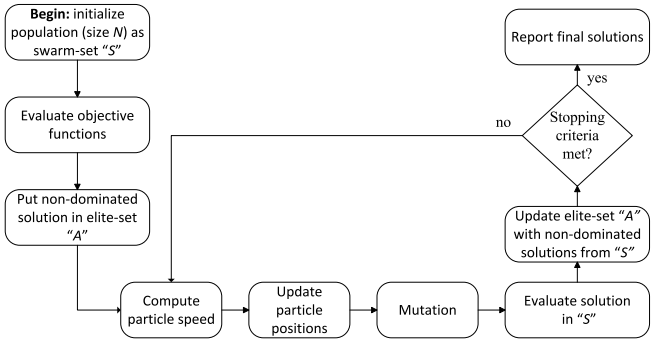


Fig. 3. Steps of the SMPSO procedure.

elitism sets is high, and can be a huge issue for large scale problems.

2) Speed-Constrained Multiobjective Particle Swarm Optimization: Particle swarm optimization (PSO), proposed by Poli *et al.* [25], is a meta-heuristic imitating the social behavior of biological creatures, such as bird flocking and fish schooling [25]. Since the first proposal, it has become a popular approach to deal with multiobjective problems. In fact, more than 30 different proposals and application cases in MOO have been reported in [26].

Among these proposals, SMPSO shows better performance based on various benchmark tests, and overcomes the limitations of other state-of-the-art EMO algorithms [27], [28]. Indeed, the benchmark evaluation in [29] shows that SMPSO is the most salient EMO technique in terms of the convergence speed toward the Pareto front with promising solution quality.

SMPSO first randomly generates a set of N initial solutions, then iteratively updates the "positions of these solutions" in the search space [29]. At each iteration, every solution adjusts its velocity to follow the local and global best solutions. An intuitive invention in SMPSO is the velocity constriction procedure. By incorporating this procedure, the solution is effectively moving through the search space [29].

The steps for solving the FPP using the SMPSO procedure are shown in Fig. 3. Initially, a set of "particles" (i.e., candidate solutions) is randomly generated, and each particle is assigned a position and a velocity in the solution space. Then, all the particles are evaluated against the objective functions. The nondominated particles will be collected and form an "elite-particle-set." In the next step, the algorithm updates the particle's position by using

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t). \tag{21}$$

The velocity for each particle $\vec{v}_i(t)$ is given by

$$\begin{aligned}\vec{v}_i(t) = w \cdot \vec{v}_i(t-1) + C_1 \cdot r_1 \cdot \left(\vec{x}_{p_i} - \vec{x}_i\right) \\ + C_2 \cdot r_2 \cdot \left(\vec{x}_{g_i} - \vec{x}_i\right).\end{aligned} \tag{22}$$

where $\vec{x}_i(t)$ and $\vec{v}_i(t)$ are the location and velocity of particle i at time t.

The $\vec{x}_{p_i}$ and $\vec{x}_{g_i}$ are the historical best and the global best points. w, C_1, C_2, r_1, and r_2 are, respectively, the inertia weight, the learning factors, and the randomness parameters for each local and global particle. The elite-set is updated with nondominated feasible solutions from each round. This step of particle updating is repeated until the terminating condition is satisfied. When the terminating condition is met, the algorithm returns the best Pareto solutions found so far.

Our preliminary experiments demonstrate that SMPSO performs efficiently on exploring the whole front. It preserves the diversity in the solutions set and always produces evenly distributed Pareto frontiers. However, after a given number of iterations, SMPSO struggles to push the solution set to converge to the true optimal front.

C. Proposed EMO Algorithm (PSONSGA)

Motivated by the disadvantages in NSGA-II and SMPSO algorithms, we propose a new EA called PSONSGA. PSONSGA is specifically designed to combine the search efficiency of the two different types of EAs (PSO and GA). More precisely, as shown in Algorithm 1, PSONSGA follows a two-phase methodology. In the first phase, the PSO procedure is executed to preprocess the Pareto solution set. While in the second phase, the NSGA-II procedure is used to reinforce the characteristic of the convergence in the solution set. More details about each phase are presented below.

PSO Phase: The PSO procedure is employed to explore the solution space, and to test different search directions (see rows 4–13). The particle updating mechanism in PSO speeds up this searching process. The diversity of the population is preserved before proceeding to the next phase. The workflow of the PSO phase is illustrated in Fig. 3. For more details about the PSO procedure, please refer to Section IV-B2.

NSGA-II Phase: In the second stage of the algorithm, the population is expected to be evenly diversified and close to the Pareto front. The purpose of this phase is not to explore the searching space but to enforce the solutions toward the optimal (see rows 16–29) [30]. The nondominated sorting in NSGA-II facilitates the concentration of the solutions. The workflow of the NSGA-II phase is illustrated in Fig. 2. For more details about the NSGA-II procedure, please refer to Section IV-B1.

In the NSGA-II phase, the aggressive selection process (see row 26) is executed to enforce the feasibility in the solution set.

Aggressive Selection Process: An offspring is included in the population only if it is nondominated or if it improves one objectives minimum/maximum value. The aggressive selection process enforces the fitness pressure upon the population. Since the diversity is preserved in the PSO-procedure, a harsh fitness process will not hurt the solutions spread character.

D. Complexity Analysis

For each iteration, the NSGA-II algorithm includes three operations, each with different time complexities.

1) Non dominated sorting with $O(M(2N)^2)$.
2) Crowding distance assignment with $O(M(2N)\log(2N))$.
3) Sorting and polling with $O(2N\log(2N))$.

Here, M refers to the number of objective functions, N represents the population size, and K is the total number of iterations. The overall complexity is (KMN^2).

Algorithm 1 PSONSGA Algorithm

Initialize a population of N individuals as "swarm population" S
Search the non-dominated solutions in the S
Put non-dominated solutions into an empty "elite archive" A
firstloop (PSO phase):
while *iteration* $\leq$ *first phase iteration* **do**
  for all $s \in S$ **do**
    Use constrained binary tournament to select a solution from elite archive
    Use the solution from last step as the global best particle
    Compute the speed of s according to the speed equation.
    Update the position of s by the speed calculated in the previous step
  Apply the polynomial mutation to 15% of the population
  Evaluate the solutions in the swarm population
  Update the elite archive: insert the non-dominated solution from swarm to archive
M $\leftarrow$ **Elite solutions in SMPSO's elite archive** *A*
C $\leftarrow$ **M (Use the solutions in M as initial population for NSGA-II)**
secondloop (NSGA-II phase):
while *iteration* $\leq$ *second phase iteration* **do**
  D $\leftarrow$ *Empty child population*
  Use constrained binary tournament to select parents in C.
  while *not enough individuals in D* **do**
    Select parent1 (by tournament selection)
    Select parent2 (by tournament selection)
    Getting child1, child2 through Binary Crossover (parent1, parent2)
    Polynomial Mutation (child1, child2)
    Evaluate child1 and child2 for their fitness values
    PSONSGA's Aggressive Selection Process
    Insert the child(ren) into D
  Execute the non-dominated-sorting over "preprocessed population" C and offsprings population D.
  Polling individuals for the next generation.
return *The set of feasible non-dominated solutions in population C*

Similarly, for each iteration, the SMPSO algorithm includes three operations.

1) Particle speed computation $O(N)$.
2) Apply polynomial mutation $O(\tau\%N)$.
3) Polling for elite set $O(MN^2)$, where $\tau\%$ is the chance of a sample solution mutating. The overall complexity for SMPSO with K iterations is $O(KMN^2)$.

Since PSONSGA comprises two optimizing phases (PSO and NSGA-II) with a total number of iterations K, we can conclude that the time complexity of PSONSGA is $O(KMN^2)$.

TABLE I
EDGE CLUSTER DEMANDS

For each edge cluster:	
Number of users within the cluster	U(10-150)
Coordinates of the edge cluster	(x, y) (within 100 x 100 Km^2)
For each user inside an edge cluster:	
Number of vCPU core	U(1-4)
Memory	U(1-40) GB
Number of packets sent per second	U(1-64)
Network access bandwidth	U(20-70) Mbps

V. EXPERIMENT RESULTS

A. Experiment Input

In this paper, we define an edge cluster to be a group of co-located clients sharing a unique IP address prefix, as is often done in practice to reduce complexity [31]. Each edge cluster has its own number of users, and for each user, the demand will be generated according to the parameters presented in Table I. This includes the vCPU, memory, and bandwidth requirements. All these demands are generated following the uniform distribution. Unlike the traditional cloud architecture, in fog network, the fog computing layer deals with the accessibility and usage of the computation resource in an ad-hoc manner. In this scenario, we believe uniformly generated requests is realistic to this context. Each edge cluster has a coordinate (x, y) which is randomly generated in the area. The Euclidean distance between the edge cluster and the fog is used to calculate the propagation delay.

In the experiment, 26 different problem sizes are solved over four different instances (i.e., we randomly generate four different problems for each size) for a total of 104 problems. The results presented below are the average over the four different instances.

B. Experiment Environment

All the experiments are run on an HP workstation with a quad-core processor, 2.66 GHz internal clock, and 4 GB of memory.

The problem inputs are generated by a program written in Java. The source codes of NSGA-II and SMPSO are taken from the JMetal framework [32]. The PSONSGA algorithm is implemented in Java.

To analyze the performance and efficiency of the various algorithms, the same problem (i.e., same users, requirements, and potential locations) is solved with four different methods: 1) the weighted sum method; 2) NSGA-II; 3) SMPSO; and 4) PSONSGA. Then, by running multiple instances for the same problem size, we average the randomness of each instance and obtain a better view of the performance of each algorithm.

C. Result Analysis

In this section, we present the experiment results to assess the performance of the above mentioned algorithms.

For the three EAs, we used a population size of 100 and the number of iterations was set to 10 000.

1) HyperVolume Indicator Comparison: The hypervolume (HV) indicator [33] is a measure used in EMO to evaluate the

searching performance. HV calculates the volume enclosed by a Pareto front approximation and a reference set. Since it overcomes the complexity arising with density-based EMO [34] and guarantees strict monotonicity in Pareto dominance [35], it has been a popular choice for evaluating EMO algorithms [36], [37]. The HV indicator examines both the convergence and diversity properties of a solution set [33]. In this paper, we use the HV indicator to examine the solution quality between the weighted sum algorithm and the three EAs.

While evaluating the HV indicator, the PSONSGA algorithm displayed its superiority in finding a good quality solution within reasonable computation time. In fact, among the 104 different problems that were solved, PSONSGA provides the best HV value for 95 problems (91.3%). NSGA-II and SMPSO each provides the highest HV value for 19 different problems (18.2%). Finally, the weighted sum approach only provides the best HV value for two problems (1.92%).

Fig. 4 provides a complete comparison of the HV values amongst the weighted sum, NSGA-II, SMPSO, and PSONSGA over the four instances with a 95% confidence interval. As can be seen, the PSONSGA algorithm achieves the best balance between convergence and diversity amongst the three EAs. The weighted sum approach produces worse HV values because each individual branch and bound search can only produce one single solution. We can also notice that for large-scale problems (i.e., problems 23, 24, 25, and 26), the HV values for NSGA-II are worse than all three other algorithms. This is due to the larger search space in large-scale problems and the concentrated effect of the nondominated sorting in the NSGA-II algorithm.

2) Inverted Generational Distance Indicator Comparison: Another widely used performance indicator is the inverted generational distance (IGD) [38]. The IGD uses a reference set to evaluate the quality of a solution set. In a nutshell, the IGD amounts to the average Euclidean distance between the solution set and the reference set [39].

In our evaluation, we use the best nondominated solutions returned by all three methods as the reference set. The average IGD values (over four instances) for each algorithm are presented in Table II. As can be seen, PSONSGA shows better results over the weighted sum method and the two existing EAs. The average IGD values for PSONSGA solutions achieve the best IGD in 25 of the 26 problem sizes. This improvement over NSGA-II and SMPSO can be explained by the exchange of information between the PSO and the GA phases in the PSONSGA procedure. This two-phase procedure has been proven to reach better Pareto front results as shown by the IGD indicator comparison.

3) Delay Gap Comparison: Since we solved the weighted sum formulation with the CPLEX solver, each solution point from the CPLEX solver is an optimal solution. In other words, under the same budget condition, the traffic delay produced by the weighted sum and CPLEX can achieve the optimum. Using CPLEX solutions as references, we calculate the average delay gaps between CPLEX and the three EAs. The average delay gaps are computed only if the same cost exists in the weighted sum solution set and CPLEX has found the optimal solutions (i.e., the relative MIP equals to zero).

TABLE II
IGD INDICATOR COMPARISON (AVERAGE OVER FOUR INSTANCES)

Problem #	Weighted sum	NSGA-II	SMPSO	PSONSGA
1	$2.16E-01$	$1.01E-02$	$2.01E-03$	$1.43E-03$
2	$1.31E-01$	$9.22E-03$	$2.09E-03$	$1.24E-03$
3	$9.81E-02$	$4.95E-03$	$1.96E-03$	$1.11E-03$
4	$4.11E-02$	$1.24E-02$	$2.46E-03$	$1.97E-03$
5	$3.57E-02$	$1.30E-02$	$3.16E-03$	$3.02E-03$
6	$3.57E-02$	$1.32E-02$	$3.55E-03$	$2.49E-03$
7	$6.12E-02$	$1.17E-02$	$3.73E-03$	$2.81E-03$
8	$9.72E-02$	$1.99E-02$	$3.68E-03$	$2.44E-03$
9	$9.83E-02$	$1.79E-02$	$4.44E-03$	$3.28E-03$
10	$1.13E-01$	$1.73E-02$	$5.32E-03$	$3.83E-03$
11	$1.34E-01$	$2.06E-02$	$4.98E-03$	$3.52E-03$
12	$1.40E-01$	$2.08E-02$	$5.54E-03$	$4.33E-03$
13	$4.47E-02$	$1.57E-02$	$2.37E-03$	$1.48E-03$
14	$5.21E-02$	$1.78E-02$	$2.59E-03$	$1.69E-03$
15	$7.05E-02$	$1.68E-02$	$2.40E-03$	$1.69E-03$
16	$1.24E-01$	$1.70E-02$	$3.11E-03$	$2.05E-03$
17	$1.18E-01$	$1.51E-02$	$3.53E-03$	$2.53E-03$
18	$1.82E-01$	$1.70E-02$	$3.50E-03$	$2.33E-03$
19	$1.70E-01$	$1.52E-02$	$3.32E-03$	$2.59E-03$
20	$3.03E-01$	$1.76E-02$	$3.43E-03$	$2.25E-03$
21	$2.62E-01$	$1.64E-02$	$3.86E-03$	$3.00E-03$
22	$3.03E-01$	$1.92E-02$	$3.61E-03$	$2.03E-03$
23	$3.48E-01$	$1.53E-02$	$3.66E-03$	$2.78E-03$
24	$4.03E-01$	$1.97E-02$	$3.34E-03$	$2.17E-03$
25	$3.68E-01$	$1.65E-02$	$3.66E-03$	$3.97E-03$
26	$5.32E-01$	$1.93E-02$	$4.43E-03$	$4.27E-03$

TABLE III
DELAY GAP COMPARISON (OVER FOUR INSTANCE SETS)

Algorithms	Min. gap (%)	Max. gap (%)	Ave. gap (%)	Std. dev. (%)	95% C. I. (%)
NSGA-II	0.00	7.80	0.30	0.81	0.30±0.15
SMPSO	0.00	8.50	0.60	1.19	0.60±0.22
PSONSGA	0.00	7.80	0.50	1.12	0.50±0.21

Table III shows the statistical analysis for the delay gaps over the four instances. The first three columns represent the minimum, the maximum, and the average delay gaps. The following two columns are the standard deviation and the 95% confidence interval for the average delay gaps. As can be seen, without considering the diversity and the distribution in the solution sets, NSGA-II gives the best delay gaps to optimal solutions. The reason behind this is that the nondominated sorting in NSGA-II concentrates the solution sets toward the optimal front. However, this sorting process sacrifices the solution diversity for a better convergence quality. This is the same reason why the solutions produced by the NSGA-II algorithm provide worse HV values. PSONSGA achieves the second best among the three EAs.

4) CPU Time Comparison: Fig. 5 provides the comparison in terms of the CPU time between the weighted sum method and the three EAs over the four instances with a 95% confidence interval. As can be seen, all three EAs can provide good quality Pareto frontier in a reasonable amount of time. For small-scale problems (i.e., problems 1 to 12), the three EAs can finish the optimization within 30 s. For large-scale problems, EAs' solution times are still within a reasonable range. For example, to solve the first instance of problem 26, NSGA-II, SMPSO, PSONSGA, each respectively, took 5303 s, 1639 s, and 5464 s. In comparison, the weighted sum method using the CPLEX solver takes 19 510 s, which is approximately 3.5 times longer than PSONSGA. We also notice that, although fast for small size problems, CPLEX's CPU time is almost increasing exponentially with respect to the problem

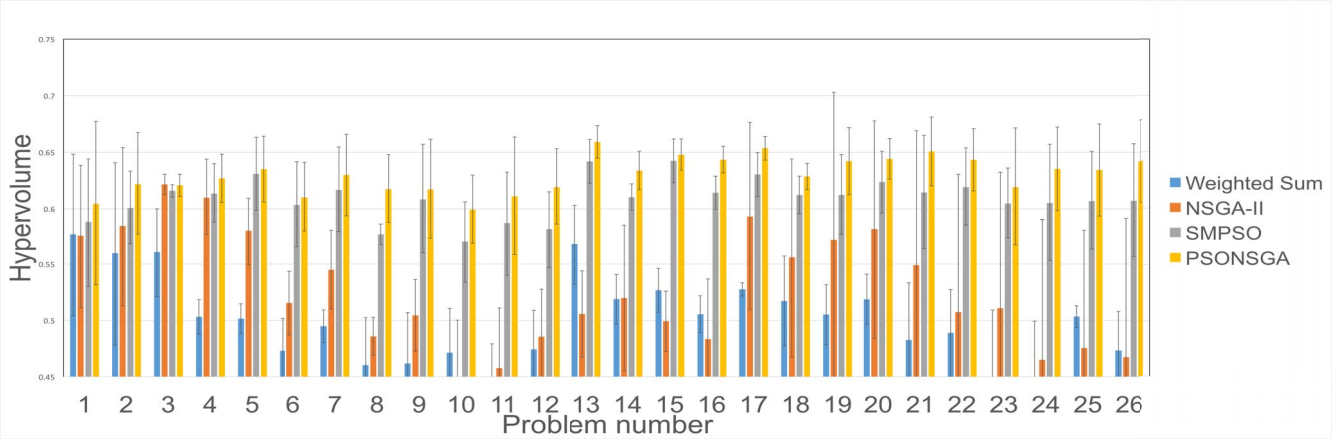


Fig. 4. HV indicator comparison (average over four instances with 95% confidence interval).

size. This corresponds with the fact that the FPP is NP-hard as described before.

Note that PSONSGA's computation time is not always higher than the existing EAs due to the time-consuming aggressive selection process. Nevertheless, with the negligible time increase, PSONSGA can achieve better experiment results which are noticeably better than existing algorithms, as shown in Figs. 4 and 5.

From the delay gap comparison shown in Table III and the CPU time comparison in Fig. 5, we can conclude that the EAs are able to use less CPU time to generate close to optimal solutions. At the same time, PSONSGA attains a good balance between the solution convergence and diversity, compared to the two existing algorithms (NSGA-II and SMPSO).

VI. Related Works

Fog computing research has attracted lots of attention and is very active area from academia and industry. Researchers have been working on studies of fog architecture [40], fog communication protocols [41] and application logic in fog networks [42]. Regarding workload allocation and resource management in fog computing networks, Dsouza *et al.* [43] proposed a policy-based resource management in fog computing, which expanded the current fog computing platform to support collaboration and interoperability between different resources. Deng *et al.* [44] focused on investigating the tradeoff between power consumption and transmission delay in the fog-cloud computing system, and they formulated a workload allocation problem that aims at minimal power consumption considering the constrained service delay. Sun and Zhang [42] proposed a fog computing structure to integrate spare resources in the network applying a crowdfunding algorithm.

Hong *et al.* [45] surveyed the existing platforms, virtualization technologies and approaches to implement the fog computing platform. The optimization model they proposed focused on maximizing the computation offloading, but failed to consider the network performance in the resource allocation process and the incurred costs in computing platform construction. Bachmann [46] proposed a fog computing framework. This framework implemented the major functionalities of a fog computing architecture. From the perspective of software engineering, this paper can be a good starting point for future research in fog computing and service deployment. Haider *et al.* [47] proposed a multiobjective model for the planning of fog networks. The model jointly optimizes network traffic and network delay. However, this model failed to consider the influence of the budget and due to its complexity, only small size problems can be solved. Liu *et al.* [48] proposed a customized optimization framework for the fog network, in which the optimization objectives can be arbitrarily combined. Compared to the heuristic algorithm, the tensor-based representation and computation model employed in this paper hugely complicated the optimization process and the time complexity of this method is unmanageable. Another closely related topic under the network planning context is the distributed-cloud planning. Hwang *et al.* [49] illustrated the planning processes of creating high-performance, scalable, reliable distributed computing systems including, the design principles, architectures, and innovative applications. Khosravi and Buyya [50] presented a taxonomy and classification of the existing techniques for resource management in achieving a green cloud computing environment. Iturriaga *et al.* [51] studied the application of multiobjective EAs for solving the energy-aware scheduling problem. The scheduling problem took account of the scheduling of large workflows in a federation of data centers. In another study, Xu and Li [52] presented a joint request mapping and response routing policy for geo-distributed cloud services. The utility functions are used to capture the performance goals. However, their model focused on the resource mapping and traffic routing in distributed cloud networks without considering the placement and sizing of the geo-distributed cloud.

Among all the work described above that are directly related to the distributed-cloud planning, none of them addresses the facility planning and provisioning in distributed computing networks as presented it in this paper. In particular, a recent comprehensive survey [12] in fog research has confirmed that work on the planning and design of fog networks deployment is vital but still lacking detailed research so far. This paper is a step toward addressing these challenges.

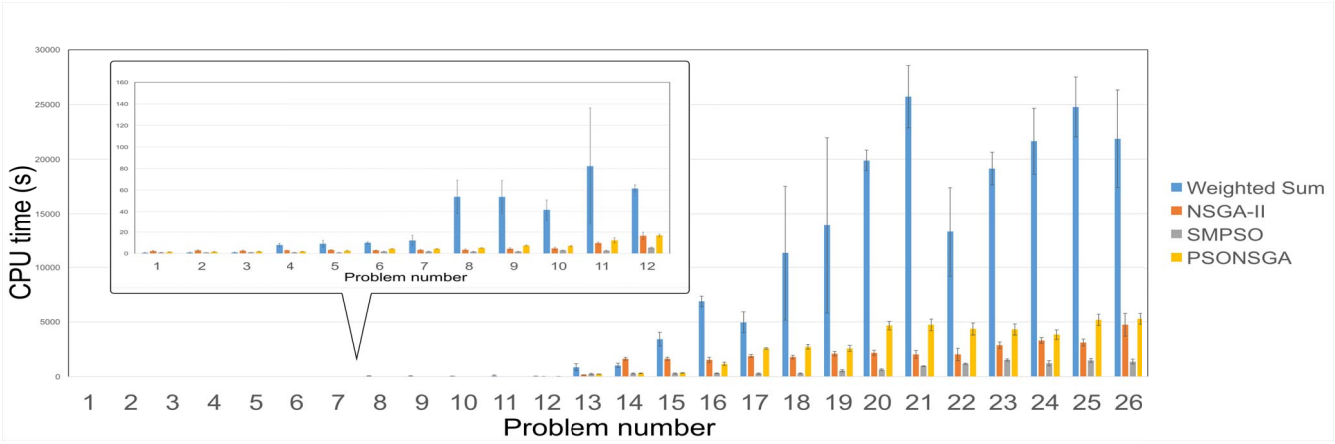


Fig. 5. CPU time comparison (average over four instances with 95% confidence interval).

VII. Conclusion

In this paper, we first proposed a multiobjective mathematical model for the planning and design of fog networks. The multiobjective combinatorial model enables us to fully explore the search domain with different fog placement, fog type selection, and user allocation strategies. The model produces the Pareto front solution set over two objectives: 1) the network delay and 2) the CAPEX. Decision makers can use this result to make reliable decision for the planning and deployment of their fog infrastructure.

To solve the proposed model, we apply one exact algorithm (i.e., weighted sum) and three heuristic algorithms (NSGA-II, SMPSO, and PSONSGA). These algorithms are compared in terms of HV indicator, IGD indicator, and delay gaps. The experiment results confirm that EAs can be notably efficient for the execution time compared to the weighted sum method. In terms of the HV and IGD indicators, PSONSGA achieves the best values for most of the problems. In contrast, NSGA-II is able to return good solutions in terms of the network performance (minimum network delay).

In the real world of FP, the number of edge clusters can be extremely large, and the network topology, interference and restrictions will be more complicated. Therefore, using the weighed sum method and CPLEX can be unrealistic. In this context, the proposed algorithm can be more appropriate. PSONSGA is recommended for solving this type of FPP as it achieves the best balance between the solution diversity, convergence, and computation time.

As future work, our model can be further extended to a dynamic expansion model in which existing fog facility can be partially closed and reopened based on the on-demand traffic. For example, some fog sites could potentially be relocated to accommodate seasonal traffic changes. PSONSGA can be used as a starting point for this direction. Another interesting avenue would be to consider the day-to-day management of fog resources. Once an initial fog network is deployed and operational, then the day-to-day management becomes an important issue. Questions like when should we use the cloud and when should we use the fog need to be addressed. Finally, the concept of green computing can also be applied to the day-to-day management where fog nodes could be turned on/off depending on the demand.

References

[1] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[2] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for Internet of Things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. New York, NY, USA: Springer, 2014, pp. 169–186.

[3] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big Data*, 2015, pp. 37–42.

[4] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[5] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398–401, Jun. 2017.

[6] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 38–43, Mar. 2017.

[7] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[8] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the Internet of Things," in *Proc. 2nd ACM SIGCOMM Workshop Mobile Cloud Comput.*, 2013, pp. 15–20.

[9] G. Orsini, D. Bade, and W. Lamersdorf, "Computing at the mobile edge: Designing elastic android applications for computation offloading," in *Proc. 8th IEEE IFIP Wireless Mobile Netw. Conf. (WMNC)*, 2015, pp. 112–119.

[10] I. Stojmenovic, "Fog computing: A cloud to the ground support for smart things and machine-to-machine networks," in *Proc. IEEE Aust. Telecommun. Netw. Appl. Conf. (ATNAC)*, 2014, pp. 117–122.

[11] J. Zhu *et al.*, "Improving Web sites performance using edge servers in fog computing architecture," in *Proc. IEEE 7th Int. Symp. Service Orient. Syst. Eng.*, 2013, pp. 320–323.

[12] C. Mouradian *et al.*, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 1, pp. 416–464, 1st Quart., 2018.

[13] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *J. Internet Services Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[14] K. Pahlavan and P. Krishnamurthy, *Principles of Wireless Networks: A Unified Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2011.

[15] M. Cvijetic and I. Djordjevic, *Advanced Optical Communication Systems and Networks*. Boston, MA, USA: Artech House, 2013.

[16] C. Feng, H. Xu, and B. Li, "An alternating direction method approach to cloud traffic management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 8, pp. 2145–2158, Aug. 2017.

[17] A. Ghosh, S. Ha, E. Crabbe, and J. Rexford, "Scalable multi-class traffic management in data center backbone networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2673–2684, Dec. 2013.

[18] N. Megiddo and K. J. Supowit, "On the complexity of some common geometric location problems," *SIAM J. Comput.*, vol. 13, no. 1, pp. 182–196, 1984.

[19] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are NP-complete," *Inf. Process. Lett.*, vol. 12, no. 3, pp. 133–137, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0020019081901113

[20] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms* (Wiley-Interscience Series in Systems and Optimization). New York, NY, USA: Wiley, 2001.

[21] 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. [Online]. Available: http://dx.doi.org/10.1109/4235.996017

[22] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization," in *Proc. ICGA*, vol. 93, Jul. 1993, pp. 416–423.

[23] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evol. Comput. World Congr. Comput. Intell.*, vol. 1, 1994, pp. 82–87.

[24] J. Ouyang, F. Yang, S. W. Yang, and Z. P. Nie, "The improved NSGA-II approach," *J. Electromagn Waves Appl.*, vol. 22, nos. 2–3, pp. 163–172, 2008. [Online]. Available: http://dx.doi.org/10.1163/156939308784160703

[25] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Jun. 2007. [Online]. Available: https://doi.org/10.1007/s11721-007-0002-0

[26] M. Reyes-Sierra and C. A. Coello Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *Int. J. Comput. Intell. Res.*, vol. 2, no. 3, pp. 287–308, 2006.

[27] S. Kachroudi and M. Grossard, "Average rank domination relation for NSGAII and SMPSO algorithms for many-objective optimization," in *Proc. IEEE 2nd World Congr. Nat. Biol. Inspired Comput. (NaBIC)*, 2010, pp. 19–24.

[28] A. Atashpendar, B. Dorronsoro, G. Danoy, and P. Bouvry, "A parallel cooperative coevolutionary SMPSO algorithm for multi-objective optimization," in *Proc. IEEE Int. Conf. High Perform. Comput. Simulat. (HPCS)*, 2016, pp. 713–720.

[29] A. J. Nebro *et al.*, "SMPSO: A new PSO-based metaheuristic for multi-objective optimization," in *Proc. IEEE Symp. Comput. Intell. Multi Criteria Decis. Making (MCDM)*, Mar/Apr. 2009, pp. 66–73.

[30] L. Magnier and F. Haghighat, "Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and artificial neural network," *Build. Environ.*, vol. 45, no. 3, pp. 739–746, Mar. 2010.

[31] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: A platform for high-performance Internet applications," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 2–19, 2010.

[32] J. J. Durillo and A. J. Nebro, "jMetal: A java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997811001219

[33] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: Optimal distributions and the choice of the reference point," in *Proc. 10th ACM SIGEVO Workshop Found. Genet. Algorithms*, 2009, pp. 87–102. [Online]. Available: http://doi.acm.org/10.1145/1527125.1527138

[34] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," in *Proc. Int. Conf. Evol. Multi Criterion Optim.*, 2007, pp. 742–756.

[35] M. Fleischer, "The measure of Pareto optima applications to multiobjective metaheuristics," in *Proc. Int. Conf. Evol. Multi Criterion Optim.*, 2003, pp. 519–533.

[36] J. Knowles and D. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 100–116, Apr. 2003.

[37] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," in *Proc. Int. Conf. Evol. Multi Criterion Optim.*, 2005, pp. 62–76.

[38] Y. Sun, G. G. Yen, and Z. Yi, "IGD indicator-based evolutionary algorithm for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8249827

[39] A. Gaspar-Cunha, C. H. Antunes, and C. Coello, "Evolutionary multicriterion optimization," in *Proc. 8th Int. Conf. EMO*, 2015, p. 591. [Online]. Available: https://books.google.ca/books?id=rtF9BwAAQBAJ

[40] B. Tang *et al.*, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proc. ASE Big Data Soc. Informat. (ASE BDSI)*, Oct. 2015, p. 28.

[41] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: Issues and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, Jul./Aug. 2016. [Online]. Available: https://doi.org/10.1109/MNET.2016.7513863

[42] Y. Sun and N. Zhang, "A resource-sharing model based on a repeated game in fog computing," *Saudi J. Biol. Sci.*, vol. 24, no. 3, pp. 687–694, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1319562X17300529

[43] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: Preliminary framework and a case study," in *Proc. IEEE 15th Int. Conf. Inf. Reuse Integr. (IRI)*, 2014, pp. 16–23.

[44] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[45] H.-J. Hong, P.-H. Tsai, and C.-H. Hsu, "Dynamic module deployment in a fog computing platform," in *Proc. IEEE 18th Asia-Pac. Netw. Oper. Manag. Symp. (APNOMS)*, 2016, pp. 1–6.

[46] K. Bachmann, "Design and implementation of a fog computing framework," M.S. thesis, Softw. Eng. Internet Comput., Vienna Univ. Technol. (TU Wien), Vienna, Austria, 2017.

[47] F. Haider, D. Zhang, M. St-Hilaire, and C. Makaya, "On the planning and design problem of fog computing networks," *IEEE Trans. Cloud Comput.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8485364

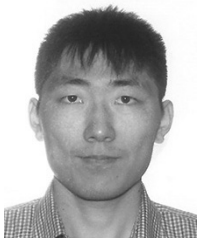
[48] H. Liu, L. T. Yang, M. Lin, D. Yin, and Y. Guo, "A tensor-based holistic edge computing optimization framework for Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 88–95, Jan./Feb. 2018.

[49] K. Hwang, J. Dongarra, and G. C. Fox, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*. Amsterdam, The Netherlands: Elsevier Sci., 2013. [Online]. Available: https://books.google.ca/books?id=IjgVAgAAQBAJ

[50] A. Khosravi and R. Buyya, "Energy and carbon footprint-aware management of geo-distributed cloud data centers: A taxonomy, state of the art," in *Advancing Cloud Database Systems and Capacity Planning With Dynamic Applications*. Hershey, PA, USA: IGI Glob., 2017, p. 27.

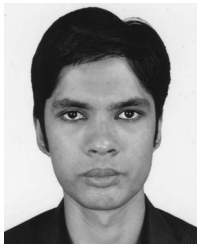
[51] S. Iturriaga, B. Dorronsoro, and S. Nesmachnow, "Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters," *Int. Trans. Oper. Res.*, vol. 24, nos. 1–2, pp. 199–228, 2017. [Online]. Available: http://dx.doi.org/10.1111/itor.12294

[52] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 854–862.



Decheng Zhang received the bachelor of engineering degree from the Department of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2013, and the master of applied science degree from the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada, in 2018.

His current research interests include fog/edge computing, fog network planning, optimization research, and cloud computing.



Faisal Haider received the bachelor of science degree in electrical and electronics engineering from American International University–Bangladesh, Dhaka, Bangladesh, in 2013 and the master of applied science degree from the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada, in 2018.

His current research interests include fog/edge computing, network planning and design, network optimization, and cloud computing.

Marc St-Hilaire (M'09–SM'13) received the Ph.D. degree in computer engineering from École Polytechnique of Montreal, Montreal, QC, Canada.

He joined Carleton University, Ottawa, ON, Canada, in 2006, where he is currently an Associate Professor with the School of Information Technology with a cross appointment with the Department of Systems and Computer Engineering. He is conducting research on various aspects of wired and wireless communication systems. With over 130 publications, his research has been published in several peer-reviewed journals and conferences. His current research interests include network planning and infrastructure, network protocols, network interconnection, and performance analysis.

Dr. St-Hilaire is actively involved in the research community. He is serving as a member of Technical Program Committees of various conferences and is equally involved in the organization of several national and international conferences and workshops.

Christian Makaya (GS'06–M'07) received the Ph.D. degree in computer engineering from the Polytechnique Montréal, Montreal, QC, Canada, in 2007.

He was a Senior Research Scientist with Telcordia Technologies, Piscataway, NJ, USA, and a Visiting Researcher with Ericsson Research, Montreal, QC, Canada. He is currently a Researcher with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. He has authored numerous technical papers in peer-reviewed journals and conferences and has filed several patents. His research has been a catalyst behind several new initiatives and technologies resulting in delivery of high-value capabilities to products and services. He leads several technical research activities in the areas of distributed systems and analytics with the mission of delivering deep technical breakthroughs. His current research interests include distributed AI and ML, edge computing, Internet of Things, network functions virtualization, policy-based management systems, and cyber-security.

Dr. Makaya was a recipient of several high-prestige internal awards by IBM for his technical contributions. He is a volunteer of the IEEE and serves on the Industry Outreach Board of the IEEE Communication Society. He served as the Co-Chair of IEEE Young Professionals for the IEEE Princeton/Central Jersey Section.