Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 2784548, 17 pages
http://dx.doi.org/10.1155/2016/2784548

Research Article

Software Defined Resource Orchestration System for Multitask Application in Heterogeneous Mobile Cloud Computing

Qi Qi,[1] Jianxin Liao,[1] Jingyu Wang,[1] Qi Li,[1] and Yufei Cao[2]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*EBUPT Information Technology Co., Ltd., Beijing 100191, China*

Correspondence should be addressed to Qi Qi; qiqi8266@bupt.edu.cn

Received 20 January 2016; Revised 21 April 2016; Accepted 22 May 2016

Academic Editor: Alessandro Cilardo

The mobile cloud computing (MCC) that combines mobile computing and cloud concept takes wireless access network as the transmission medium and uses mobile devices as the client. When offloading the complicated multitask application to the MCC environment, each task executes individually in terms of its own computation, storage, and bandwidth requirement. Due to user's mobility, the provided resources contain different performance metrics that may affect the destination choice. Nevertheless, these heterogeneous MCC resources lack integrated management and can hardly cooperate with each other. Thus, how to choose the appropriate offload destination and orchestrate the resources for multitask is a challenge problem. This paper realizes a programming resource provision for heterogeneous energy-constrained computing environments, where a software defined controller is responsible for resource orchestration, offload, and migration. The resource orchestration is formulated as multiobjective optimal problem that contains the metrics of energy consumption, cost, and availability. Finally, a particle swarm algorithm is used to obtain the approximate optimal solutions. Simulation results show that the solutions for all of our studied cases almost can hit Pareto optimum and surpass the comparative algorithm in approximation, coverage, and execution time.

1. Introduction

Along with the development of cloud computing, the technologies such as mobile device, wireless network communication, pervasive computing, and the technology cloud computing influence each other. This results in the mobile cloud computing (MCC) that takes heterogeneous access network as the transmission medium and uses mobile devices as the client and becomes the newest evolution trends of mobile computing [1]. Nowadays there exists three types of heterogeneous resources in MCC, which coexist and are overlapped with competition and interdependence, depicted in Figure 1.

(1) Infrastructure-based: mobile device acts like a thin client connecting over to the remote server through wireless network. The application is deployed on the multiple cloud computing infrastructures that are belonging to different cloud providers, located in various areas, and have different rental cost and available resources, for example, Amazon EC2 and Google Compute Engine. From the user's requirement and Quality of Service (QoS), the suitable composition of cloud computing resources can realize the optimization of resource provision.

(2) Cloudlet-based: mobile device offloads its workload to a local "cloudlet" comprised of several multicore computers with connectivity to the remote cloud servers [2]. Consequently, the mobile cloud computing reduces an execution time of mobile applications and the energy consumption of mobile devices. For example, the smart router in the bus can provide cloud computing resource for passengers. At the same time, cloudlet servers can be the Agent between mobile device and infrastructure-based cloud computing resource, which execute strong interactive and delay sensitive services with on-demand payment, such as online game and virtual reality [3].

(3) Mobile P2P resource pool: a group of mobile devices acts as both the resource providers and resource consumers and makes up a mobile peer-to-peer network [4]. They

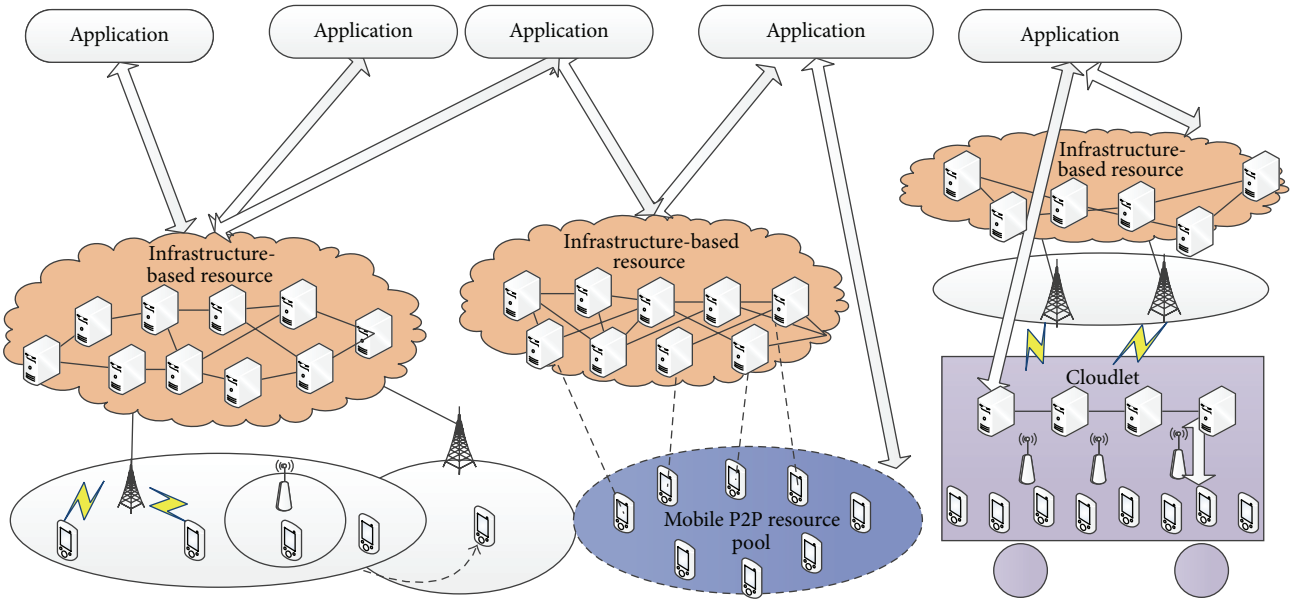


Figure 1: Mobile cloud computing scenario.

provide accessing to local or Internet based cloud services for other mobile devices, share their resources, and execute services cooperatively. Moreover, the connected mobile device can offload part of the application to the above (1) and (2) resources by setting clone nodes, which constructs the multilayer MCC resources pool [5]. Currently, fog computing proposed by Cisco as an extend concept of cloud computing pushes the edge computing paradigm up to the end users' terminals (e.g., smart phones) and other devices that are able to store pieces of data and execute service components locally [6]. Thus, fog computing is similar to mobile P2P pool, in which the data, processing, and application are deployed at edge devices instead of the cloud.

From these three types of resources, the energy-constrained mobile device plays an important role in MCC. The energy consumption as well as users' mobility is expected to vary during services invocation, which leads to uncertainty of resource provision. Furthermore, the mobile resources have more heterogeneity and dynamic comparing with the traditional cloud computing, which results in huge difference in usability.

Along with the users' requirements becoming richer, many emerging cloud applications are complex combinations of multiple tasks and require predictable performance and high availability [7]. For example, to satisfy a travelling photos search requirement, an application that consists of five tasks is generated. *Step 1.* Render photo. This task collects photos when users are travelling and renders them to the required size and quality. *Step 2.* Collect environment information. This task collects the location, height, and temperature information that is used to help search the landscape picture. *Step 3.* Feature extraction. The features of uploaded photo are extracted based on color, texture, and shape. Then the high dimensional feature vector is mapped into hash value. Steps 2 and 3 can be executed in parallel. *Step 4.* Search photo. By taking use of the hash value and the related environment information, this task searches the similar photos in the data center. *Step 5.* Translate explanation words. When receiving the photo search result, this task returns the photo results and translates the corresponding explanation word to the users' native langue. In this example, each task has its own computation, storage, and bandwidth requirement. The resource allocation as well as the results dependency relations should all be considered when designing the resource orchestration.

Traditionally, the heavy computing applications migrated from mobile device are usually deployed in one cloud computing resource. Nevertheless, for the complicated application, the multiple tasks may be located in different cloud computing resource. When users offload the tasks for improving capability and performance, heterogeneous MCC resources provide multiple choices for users. Thus, MCC resource management requires integrating and considering multiple aspects of the resources, such as resource location, intercommunication between tasks, and resource capability, aiming at east mobility effect on application performance and energy consumption. Ranjan et al. define resource orchestration as the set of operations that cloud providers and application providers undertake for selecting, deploying, monitoring, and dynamically controlling the configuration of hardware and software resources as a system of QoS-assured components that can be seamlessly delivered to end users [8]. For the heterogeneous MCC resources that lack integrated management and can hardly cooperate with each other, how to orchestrate the multiple resources automatically by meeting QoS objectives of both application provider (e.g., maximizing availability and throughput, while minimizing latency and avoiding overload) and resource providers (e.g., maximizing usage and minimizing energy) is a challenge problem [9].

To realize a power-aware architectures and programming resource provision for heterogeneous energy-constrained computing environments, a software defined resource orchestration is proposed. The framework contains a novel Resource Controller and an algorithm that solves the fore-mentioned problem, aiming at achieving maximal performance experienced by the end users and minimal cost in cloud resources favored by cloud providers. The main contributions of this paper are described as follows:

(1) Make the heterogeneous MCC resources be dynamic allocated. Taking use of the Software Define Networking (SDN) theory to decouple mobile resource control from all user plane elements, we are actually able to create a software defined system with a programmable central control node being charge of distributed resource flexible scheduling. In addition, the proposed resource orchestration can be also applied in the fog computing by taking the capability of edge devices as resources, just like the mobile P2P pool.

(2) Define the resource orchestration problem and obtain the approximate optimal solutions in acceptable time. Since the multiple tasks contain dependency relations and heterogeneous MCC resources have different location, computation capability, resource availability, and energy consumption, the resource orchestration can be transformed to the service selecting problem. Solving this problem, we explore the heuristic algorithm and intelligent optimization to make MCC resources utilization fairly and save mobile device energy. To the best of our knowledge, it is the first work to study the resource orchestration problem for multiple tasks for achieving multiobjective optimization of resource utilization.

(3) Consider mobility features by taking energy consumption and resource availability as two of the multiobjective. These objectives better suited the needs of mobile scenario compared to traditional cloud computing. Importantly, a mobility model is included in the proposed MCC resource orchestration system. By considering users' handoff times, moving direction, and their locations during mobile application running, the availability for three types of MCC resources is calculated.

In Section 2, we first describe the related work. We then present the system architectural design in Section 3. The resource orchestration in MCC problem and solutions are presented in detail in Section 4. Numerical evaluation and real world experiment are presented in Section 5. Section 6 concludes the paper and presents the future work.

2. Related Work

Researchers focus on MCC mainly including resource provision, application offload, and migration. The relevant work for resource orchestration is application offload, application partition, and resource scheduling.

When offloading, the application codes are transmitted through the wireless network between local device and remote resource node [23], so that user's mobility and network bandwidth impact on the offload result [24]. Thus, the key point of offload is deciding when the application model moves outside the mobile device. The deciding procedure is very complicated, which is determined by multiple factors, such as user's perfective, network status, capability of mobile device, service type, and resource property. Moreover, the deciding for application migration among different resource areas is the same with the offload deciding [25], which keeps the goal of solving optimization problem with some constrictions and maximizing the revenue of mobile device and cloud computing systems. These multiobjective and multiconstriction problems are usually analyzed by theories such as noncooperative gaming, cooperative gaming, Markov decision process, graph theory, and solved by the optimization algorithms.

(1) Noncooperative gaming is as follows: formulating the offload problem as the noncooperative gaming between mobile device and resource providers and analyzing the strategy by introducing the incentive and punishment mechanisms. Ge et al. [10] formulate energy minimizing problem as noncooperative gaming and propose an effective algorithm that can achieve Nash equilibrium in polynomial time. Wang et al. [11] consider offload destination place when making offload decision and then design a two phrase noncooperative gaming that can be solved by convex optimization.

(2) Cooperative gaming is as follows: deciding whether the resource providers should share resource and calculating the optimized quantity based on the revenue sharing idea [12]. Misra et al. [13] reduce migration cost and energy consumption by appropriate incentive strategy and guarantee the resource providers obtaining a fair revenue based on the Shapley value.

(3) Markov decision is as follows: considering the mobile device and MCC resource providers as a whole system whose status changes according to the offload and migration. Gabner et al. [14] take use of the Markov decision processing to evaluate the performance effect caused by offload for mobile device and MCC resource. According to this, offload strategy is proposed in order to avoid the service failure caused by unstable wireless network and maximize the service successful probability. Liang et al. [15] take energy consumption and resource cost into consideration and maximize the system revenue by the appropriate resource allocation. The semi-Markov decision model is used to analyze the application migration among mobile device and several cloud computing resources.

(4) Graph theory is as follows: abstracting the application offload to a weighted graph and finding the solutions by optimization algorithms. Deng et al. [16] propose a service offload system where multiple mobile services in workflows can be invoked to fulfill their complex requirements. It solves the problem whether the services of a workflow should be offloaded or not aiming to optimize execution time and energy consumption of executing mobile services, with the consideration of dependency relations among component tasks. Moreover, a genetic algorithm based offloading method is designed and implemented.

Sometimes, the resource orchestration requires dividing the application into multiple tasks. Thus, a fine application partition is the precondition of resource orchestration. The application partition can be formulated as an optimization problem with different objectives, such as less interactive

Table 1: Comparison of related approaches.

Aspects	Reference	Methods	Contain multitask dependency	Consider resource difference	Contain multiobjective	Consider mobility
Deciding offloading	[10, 11]	Noncooperative gaming	No	Yes	No	No
	[12, 13]	Cooperative gaming	No	Yes	No	No
	[14, 15]	Markov decision	No	Yes	Yes	Yes
	[16]	Graph theory	Yes	No	No	Yes
Application partition	[17]	Graph theory	Yes	Yes	No	No
	[18]	Genetic algorithm	Yes	No	No	No
Resource scheduling	[19]	Directed acyclic graph	Yes	Yes	No	No
	[20]	Self-adaptive learning PSO	Yes	Yes	No	No

time, less commutation cost, lower memory occupation, and several constraining conditions such as tasks dependency, delay, and bandwidth [4]. It is proved that this problem is NP-complement with multiple approximate solution space [26], which can be solved by genetic algorithm, simulated annealing algorithm, and other optimization algorithms. Verbelen et al. [17] propose a static partition algorithm based on the graph theory, which distributes the multiple modules of complicated application to the different configured servers in cloud. The authors consider the resource load balance and try to reduce the communication cost among modules. Yang et al. [18] focus on how to optimize the computation partitioning of a data stream application between mobile device and cloud to achieve maximum throughput in processing the streaming data. A framework that provide runtime support for the dynamic computation partitioning and enable the partition result to share between users is proposed, which contains a genetic algorithm for optimal computation partition.

The resource orchestration is similar to the task scheduling. For resource orchestration in the cloud computing, Horizon Project [27] which firstly points the heterogeneous cloud computing resource from multiple providers should be composited as a services to enhance availability, flexibility, and elasticity and to meet targeted performance constraints. Bittencourt et al. [19] divide the application into dependent modules and make use of the directed acyclic graph to allocate cloud computing resource for each task. The authors introduce the scheduling algorithms in hybrid clouds to reduce the execution time. Zuo et al. [20] propose a resource allocation framework to take use of the private cloud and the external public cloud together when its own resources are not sufficient to meet the demand. The problem is how to allocate users' tasks to maximize the profit of IaaS provider while guaranteeing QoS. It is formulated as an integer-programming model and solved by a self-adaptive learning particle swarm optimization algorithm.

In conclusion, computation offloading can improve capability of mobile services through migrating heavy computation tasks to powerful servers in clouds. For the complicated application in MCC, resource difference, multitasks dependency, multiobjective, and users' mobility should all be considered when orchestrating the resources. The pros and cons of the above approaches, including their methods and aspects, are compared in Table 1. For example, the work in [16] solves the problem of multitask offload decision, without any consideration of the difference of MCC resource. Moreover, the work in [18] only takes into account the resource allocation and task execution between mobile device and cloud infrastructure, which ignores the moving speed and direction. The work in [19] only focuses on the resource allocation without considering users mobility and their cooperation. None of the current works focus on the four aspects of resource provision. Therefore, we focus on the resource orchestration by taking mobile device moving path, energy consumption, and resource cost into consideration and design a software defined system that controls the heterogeneous MCC resources.

On the other hand, the games theory and other approaches from certain perspective of resource provision are used in previous work [10–16], which consider single objective or make multiobjective to a weighted single objective. Generally, the games theory and Markov decision process are used to analyze problem, while the solutions are obtaining by various heuristic algorithms. For example, considering our multiobjective resource orchestration, games theory should take the metrics energy consumption, resource charge, and availability as players, which is very different with the previous works where the mobile devices are assumed as players. Actually, the three metrics do not affect each other directly which is not suitable for using game theory, and the difficulty of formulation increases. Hence, we formulate the resource orchestration as a multiobjective problem and propose a particle swarm algorithm to find the appropriate solutions provided for users.

3. System Architecture

In this section, we first introduce our system model along with software defining idea. Second, we introduce the system functionalities. Then, we define the interactive protocol between the mobile device and heterogeneous MCC resource providers.

3.1. Software Defined Infrastructure. Recently, SDN has emerged, which introduces a logically centralized network with a programmable controller and flow-based switches. It focuses on control of network routing and switching and touches on the heterogeneous and multiprovider network

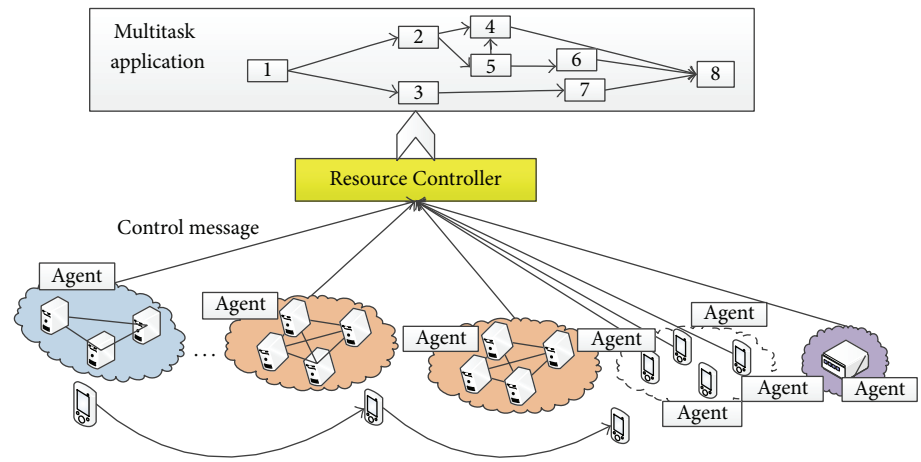


Figure 2: SDI-based resource orchestration system.

management. The openness of SDN enables the network resource to be used flexibly by the applications. Accordingly, the cloud computing infrastructure with some open resource management functions becomes increasingly programing, which is referred to as a software defined environment [28]. We extend the concepts of software defining into the realm of MCC resource management and combine of the optimization functionality that allow for deploying the complicated application to the distributed heterogeneous resources. A Software Defined Infrastructure (SDI) based MCC system is proposed, which contains virtualization of compute, network, and storage as well as dynamic resources provision through programed orchestration. The centered resource management is conducive to the interaction among the heterogeneous resources. The dynamic scheduling of heterogeneous resources for multitask application is separated from the resource providers.

As shown in Figure 2, the SDI-based system uses centered Resource Controller for resource orchestration as well as distributed flexible Agents for interaction. The scheduling of heterogeneous resources for multitask application is separated from the resource providers. The centered Resource Controller is conducive to the collaboration among the heterogeneous resources, which is placed between the heterogeneous MCC resources and the mobile devices. The communication and monitoring functions for the resources are encapsulated in the Agents, which is easy to install.

3.2. System Functionalities. For realizing the resource orchestration in heterogeneous mobile cloud computing environment, the Resource Controller for resource orchestration contains four functions, depicted in Figure 3. Taking use of the information provided by service provider, cloud resource provider, and the mobile user, the system decides which resources will be used to execute the application. The interactive and information sharing mechanism in the system guarantees the functions which cooperate with each other.

Monitor function collects and aggregates the status and performance metrics of mobile cloud resources distributive. By this method, the controller obtains the request number that the resource provider has accepted and collects the related sensor data like location, temper, and so on.

Analysis function takes use of the monitoring information and abstracts the resource orchestration as an optimal problem. The application partition is used to divide the complicated applications if the application itself does not contain several separated tasks. Then according to resource management requirement and the calculating results, this function provides the optimization of resource allocation strategies for an application aiming at realizing the optimized resource allocation in the heterogeneous mobile cloud resource, that is, cloud computing infrastructures, cloudlet, or the neighboring cloud resource nodes.

Executing function fetches the analysis results from the information center, migrates the service to corresponding resource area, and controls the task execution. The service deployment and migration procedure can consult [29, 30].

Information center is a database communicating with the above three functions. It gathers the perceiving and monitoring information from the distributed resource areas and transmits the resources orchestration policy from the analysis function to executing function.

Agent is deployed in each resource domain to monitor the available virtual instances and embedded in the mobile device in order to gather the moving speed, network status, resource utilization, and remaining battery energy. On the other hand, the Agent is used to transmit the task execution results to the next MCC resource area according to the orchestration results.

3.3. Interaction Procedure. The heterogeneous cloud computing resources may not belong to the same provider. An appropriate incentive-compatible policy such as cooperative game theory for multiple cloud providers [12] makes the resource provision among different cloud providers practical. Based on this, our goal is improving the application performance by offloading each task to the appropriate resource. We assume that Service Level Agreement (SLA) of cloud resources is determined, so the cost and energy consumption are set before orchestration.

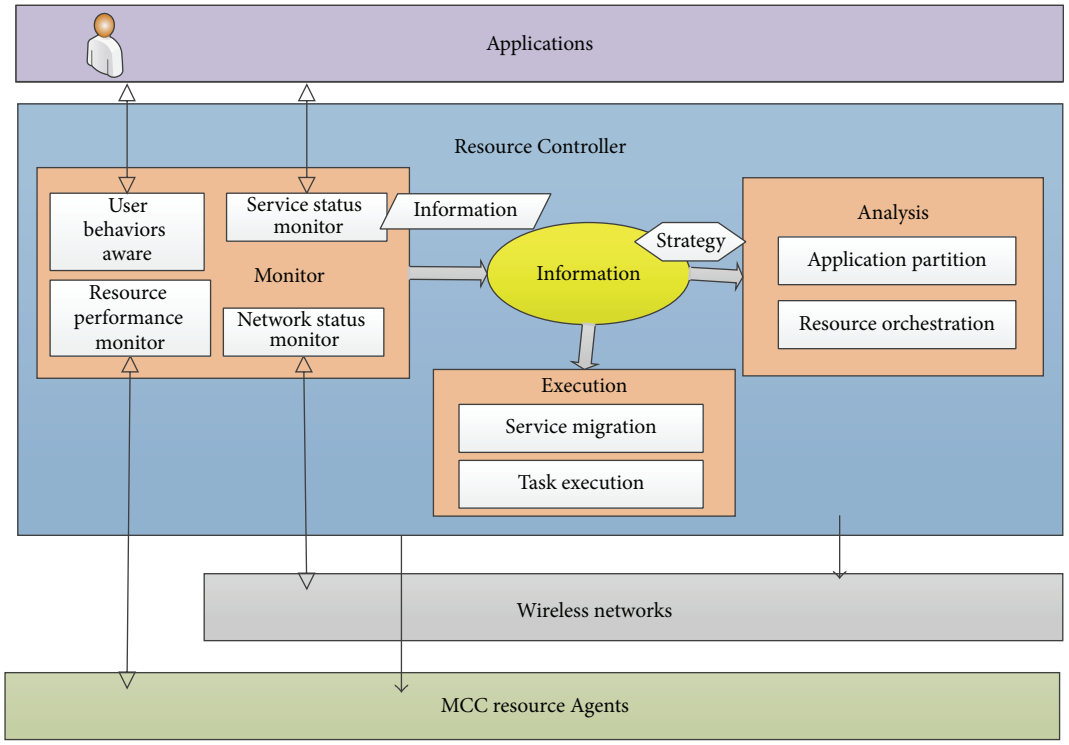


Figure 3: Resource orchestration controller.

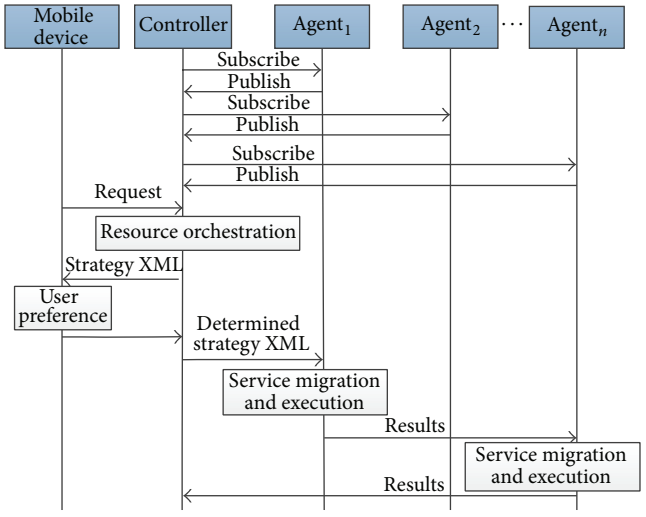


Figure 4: Interaction procedure in SDI.

The SDI resource orchestration is depicted on Figure 4. For collecting the resource status, Monitor function in Resource Controller subscribes the information from each Agent. When the user's location changes or the timer is triggered, Agent sends the information to the Monitor function, and then it is pushed to the information center. In this way, the Monitor function can keep track of the performance metrics of the cloud resources. When a user's request arrives at the Resource Controller, the analysis function finishes the resource orchestration and transmits the strategies to the mobile device by the execution function. As there are multiple objects for the orchestration, several solutions may be provided to users. User chooses one orchestration strategy and sends it to the Resource Controller. The strategy is depicted by XML format that includes the identification of resource, task, and depiction of resource requirement.

When the first MCC resource Agent receives the strategy, it performs the service migration and executes the task. After that, it deletes the records of itself and transmits the results to the next MCC resource Agent. The Agent collects the results of its previous tasks, updates the results, and transmits the results as well as the strategy to the next Agent. The interaction messages of Agents and Resource Controller are implemented by using Representational State Transfer (REST) Application Programming Interfaces (APIs) that is easy for extending.

4. Optimal Resource Orchestration

In this section, we mainly focus on the MCC resource planner of the above framework and propose orchestration strategies to reduce energy consumption and resource charge as well as increase resource availability for running mobile applications by a multiobjective optimization algorithm.

4.1. Problem Definition. The heterogeneous MCC resources are abstracted to several services with specific functions and parameters. The multitask in a complicated application is modeled as a specific dataflow graph and denoted by directed acyclic graph (DAG), depicted in Figure 5. Due to the difference existing in the MCC heterogeneous resources, the

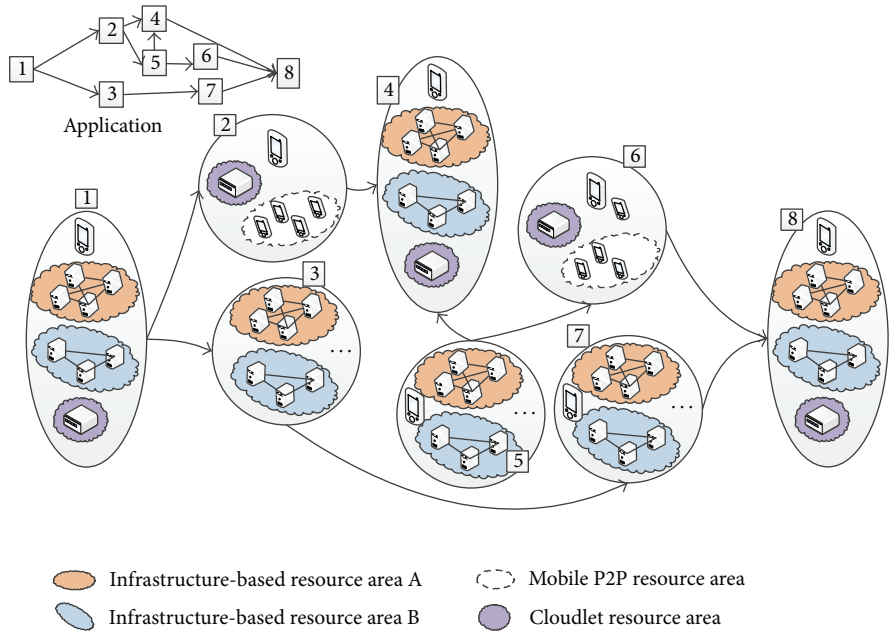


Figure 5: Example of resource orchestration in MCC.

appropriate resource service is chosen for each task according to the global optimization goal during users' movement. As the metrics such as energy consumption, resource cost, and resource availability are not relevant or not following the same trend or even conflicting with each other, improving one metric may result in the deterioration of another one. We try to use the theory of multiobjective optimization to solve the resource orchestration problem.

The multiobjective resource orchestration is to find a solution set *Resource Orchestration Path* ($\mathrm{ROP}_{\mathrm{set}}$) in which each solution ROP_z has extreme optimal value of aggregated metrics; that is, ROP_z should fulfill

$$\begin{aligned}&\min F\left(\mathrm{ROP}_z\right)\\&\quad=\left[\min E\left(\mathrm{ROP}_z\right),\min C\left(\mathrm{ROP}_z\right),\max R\left(\mathrm{ROP}_z\right)\right],\end{aligned}\tag{1}$$

where $\mathrm{ROP}_z \in \mathrm{ROP}_{\mathrm{set}}$. $E(\mathrm{ROP}_z)$, $C(\mathrm{ROP}_z)$, and $R(\mathrm{ROP}_z)$ are the energy consumption, resource charges, and resource availability of SCP_d respectively. F is the objective vector and function.

Energy consumption ($E_T^{i,r}$) means the total energy consumption of mobile device when task i is offloaded to the resource r. According to the energy consumption model for MCC applications in [31], we choose the most important factors, that is, wireless access technology and the mount of the transmitted data, to define the energy consumption value that is more convenience to gather the parameters. E_{trans}^r is the energy consumption of transmitting per unit (bit) data to resource area r. V_{offload}^i is the data volume of migrating the task i to MCC and $V_{\mathrm{interaction}}^i$ is interaction data volume between the mobile device and cloud resource during the task execution. E_{comp} is the energy consumption of computing per unit (bit) data and V_{comp}^i is the local data volume of task i:

$$E_T^{r,i}=E_{\mathrm{trans}}^r\left(V_{\mathrm{offload}}^i+V_{\mathrm{interaction}}^i\right)+E_{\mathrm{trans}}\cdot V_{\mathrm{comp}}^i.\tag{2}$$

Resource charge ($C_T^{i,r}$) refers to the price pay for the specific MCC resource r where task i will be offloaded. As resource charges is an open-ended field, resource providers can define their own policy. Hence, a common charge policy is presented. Let C_{comput}^r be the computing charge of selected MCC resource r, let $C_{\mathrm{commun}}^{r,r'}$ be the data communication cost from resource r to resource r', where task j will be offloaded, and let C_{trans}^a be the transmitting charge per unit (bit) data by access network a. V_{commun}^{ij} is the data volume transmitted from task i to task j:

$$C_T^{i,r}=C_{\mathrm{comput}}^r+C_{\mathrm{commun}}^r V_{\mathrm{commun}}^{ij}+C_{\mathrm{trans}}^a\cdot V_{\mathrm{interaction}}^i.\tag{3}$$

Resource availability ($R_T^{i,r}$) means the resource availability of resource for task i when a mobile device is moving to a confirmed direction. Li et al. [1] point out that unavailability of MCC is resulted by the heavy workload, insufficient network bandwidth, and long latency. The latter two factors are impacted by user's mobility. As user's mobility has different impact on the three types of MCC resource, we calculate the resource availability for infrastructure-based resource, cloudlet, and mobile P2P resource pool, respectively.

(1) Infrastructure-Based Resource. The MCC environment consists of multiple cloud service domains partitioned by geographic locations. One domain usually provides resource to local mobile devices that are connected through local

base stations or Internet access points [15] and the service may be migrated to a new domain according to its moving [30]. Therefore, considering the resource geographical location, the number of traversed access networks during users' movement affects the resource usability. Assume that the execution time of task i is exponentially distributed with mean value of μ_i^{-1}. The mean residence time in one access network η^{-1} is the general continuous random variable with the probability density function of $f_{\text{res}}(x)$. Assume that $f_{\text{res}}^*(x) = \int_0^\infty f(x)e^{-\lambda x}dx$ is the Laplace-Stieltjes Transform for the $f_{\text{res}}(x)$, and then $1/\eta = \int_0^\infty x f_{\text{res}}(x)dx$. Assume that the number of handoff times during a task execution is N_{handoff} that can be deduced according to [32]:

$$N_{\text{handof}}^{i,r} = \sum_{k=0}^{\infty} kP(k) = \sum_{k=1}^{\infty} \frac{k\mu_i}{\eta} \left[1 - f_{\text{res}}^*(\mu_i)\right]^2 \left[f_{\text{res}}^*(\mu_i)\right]^{k-1} = \frac{\eta}{\mu_i}. \tag{4}$$

González et al. [33] present that human mobility shows a very high degree of temporal and spatial regularity and that each individual returns to a few highly frequented locations with a significant probability. The distribution of movement distance D_{mv} follows a truncated power-law $P(D_{\text{mv}}) = (D_{\text{mv}} + D_0)^{-\beta} \exp(-D_{\text{mv}}/k_{\text{mv}})$. Here, the parameters β, k_{mv}, and D_0 can be set according to [33] and the users' historic data.

Accordingly, the mean residence time in one access network η^{-1} can be calculated by

$$\eta^{-1} = \frac{d_{\text{bs}} T_{\text{observ}}}{E(D_{\text{mv}})}, \tag{5}$$

where d_{bs} is the diameter of the area overlapped by a base station, T_{observ} is the observed time of history dataset, and $E(D_{\text{mv}})$ is the expectation of moving distance.

Then $R_T^i = U_r / N_{\text{handoff}}^i$, where U_r is the resource usability of MCC resource r. U_r can be calculated by historic data for each resource: $U_r = T_{\text{available}}^r / T_{\text{total}}^r$, where $T_{\text{available}}^r$ and T_{total}^r are the normal operation time and total observation time of resource r, respectively.

(2) Cloudlet Servers. For the resource server is moving together with the mobile device, $R_T^i = U_r$.

(3) Mobile P2P Resource Pool. As the shared resource among mobile devices is not stable as users are moving when they cooperate with each other, the headlight model [34] is used to calculate the availability of resource node. In Figure 6, the grids denote the cooperative mobile devices, and the resource available zone is a virtual fan-shaped area along the direction of user movement similar to the headlight of a vehicle. The luminance of headlight is in proportion to the distance and is inversely proportional to angle [34], which is indicated by the following formula: $\alpha = k|\cos\theta|/r^2$. Here, k is an unable constant that can be used to control the intensity of the headlight. To obtain the available intensity R_{np} of each sharing node, we approximate the virtual luminance, as illustrated in Figure 6(b), and calculate the luminance of the headlight overlapping area. We partition the headlight zone into smaller grids and compute the luminance of each grid. Since each grid is in a regular shape with the angle from θ_s to θ_e and the distance from l_s to l_e, the luminance can be easily computed by the following formula:

$$R_{np} = \sum_{g \in G} I_g(\theta_s, \theta_e, l_s, l_e), \quad \text{where } I_g(\theta_s, \theta_e, l_s, l_e) = \int_{\theta_s}^{\theta_e} \int_{l_s}^{l_e} \alpha\gamma \, d\gamma \, d\theta. \tag{6}$$

G is the set of grids, and the available intensity of the mobile node is the sum of the luminance of overlapping grids of user's moving direction. Thus, we determine $R_T^{i,r} = U_r \cdot R_{np}$.

There are four basic topologies in MCC resource orchestration, sequence, parallel, selective, and loop. These topologies are able to construct the vast majority of composited applications. Assume M tasks in a mobile application. I_L^i is the number of loops for task i and P_S^i is the probability of task i which is selected in the selective structure. The value of objective metrics, energy consumption, and resource cost can be calculated as follows:

$$E(\text{ROP}_z) = \sum_{i=1}^{M} E_T^{r,i} L^i P_S^i, \qquad C(\text{ROP}_z) = \sum_{i=1}^{M} C_T^{r,i} I_L^i P_S^i. \tag{7}$$

Let t' be one of the task routings in the selective structure, let $n_{t'}$ be the number of tasks in one task routing, let T_c be a task routing set in one selective structure, and $P_S^{t'}$ is the probability of task routing t' which is selected in the set T_c. Assume that there are C select structure for an application; the resource availability of orchestrated resource is

$$R(\text{ROP}_z) = \prod_{i=1, t_i \notin \bigcap_{c=1}^{C} T_c}^{M} R_T^{r,i} I_L^i \cdot \prod_{c=1}^{C} \left(\sum_{t' \in T_c}^{T_c} \prod_{j=1}^{n_{t'}} R_T^{r,j} \cdot P_S^{t'} \right). \tag{8}$$

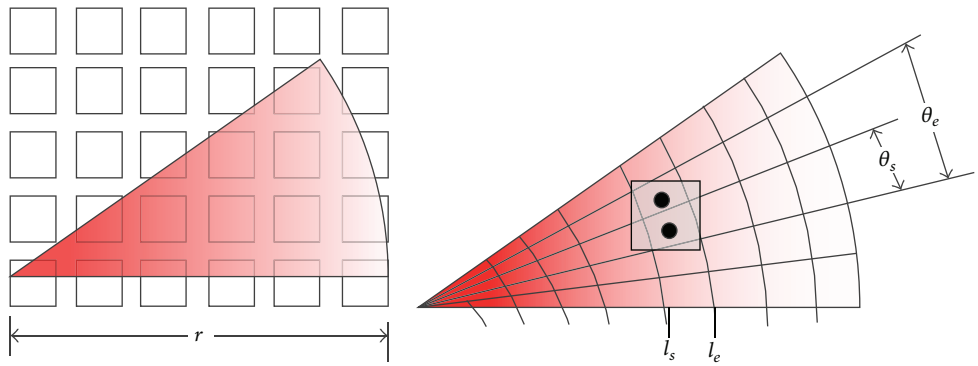
The resource orchestration is to select the optimal from all possible candidates considering multiple objectives (e.g., energy, cost, and availability). This problem is known as multiobjective optimization problem. Compared to single objective problems, a multiobjective problem has multiple optimization goals that increase the complexity enormously. To facilitate descriptions, we present some definitions.

Definition 1. Solution ROP_0 dominates ROP_1, denoted as $\text{ROP}_0 \succ \text{ROP}_1$, if and only if

$$\left(f_g(\text{ROP}_0) \le f_g(\text{ROP}_1), \forall g \in \{1,2,3,4\}\right) \wedge \left(f_h(\text{ROP}_0) < f_h(\text{ROP}_1), \exists h \in \{1,2,3,4\}\right). \tag{9}$$

Definition 2. Solution ROP_0 is a Pareto optimum, if and only if

$$\neg \exists \text{ROP}_1 : \text{ROP}_1 \succ \text{ROP}_0. \tag{10}$$



(a) Headlight model (b) Luminance calculation of headlight overlapped area

FIGURE 6: Headlight model for availability of mobile P2P resource.

Definition 3. The set of Pareto optima is defined as

$$\mathrm{ROP}_{\mathrm{set}} = \left\{\mathrm{ROP}_z \neg\exists \mathrm{ROP}_{z'} \succ \mathrm{ROP}_z\right\}. \tag{11}$$

Definition 4. The corresponding objective value area P_F of all Pareto optima is defined as the Pareto front:

$$\begin{aligned} P_F = &\left\{F\left(\mathrm{ROP}_z\right)\right. \\ &= \left(E\left(\mathrm{ROP}_z\right), C\left(\mathrm{ROP}_z\right), 1/R\left(\mathrm{ROP}_z\right)\right) \mid \mathrm{ROP}_z \\ &\left.\in \mathrm{ROP}_{\mathrm{set}}\right\}. \end{aligned} \tag{12}$$

Furthermore, the multiobjective function min $F(\mathrm{ROP}_z)$ is extensible, which can be added more objectives. Some new formula can also be included in the multiobjective function. For example, if an application is delay sensitive and the performance of task execution is important for selecting resources, the task execution delay can be considered as one of the objectives. Alternatively, as the task execution delay is also affected by the data volume V^{ij}_{commun} and $V^{i}_{\mathrm{interaction}}$, C^{r}_{comput}, $C^{r,r'}_{\mathrm{commun}}$, and C^{a}_{trans} can be replaced by corresponding parameters of delay, and the resource charges can be transformed to a performance metric, that is, task execution delay in the selected resources.

4.2. Denotations in the Problem. The denotations in resource orchestration are shown in Figure 7. When request is forwarded to the Resource Controller, it can be denoted by a task set $T_{\mathrm{Set}} = \{T_1, \ldots, T_M\}$, where $M = 5$. Then Resource Controller maps the task set to corresponding MCC resources and determines each task's MCC resource candidates $R_{\mathrm{Set}}i = \{RS_1, \ldots, RS_m\}$, $i = 1, \ldots, M$. The Resource Controller chooses one resource service RS_j for each task and forms a candidate orchestration $RS = \{T_1RS_3, \ldots, T_4RS_6, T_5RS_1\}$.

Considering that the Multiobjective Particle Swarm Optimization (MOPSO) has the properties of less parameter and easier implement comparing with the other regression or bio-based algorithms [20, 35], we take use of the MOPSO to solve this problem. The advantages of MOPSO comparing with Nondominated Sorting Genetic Algorithm II (NSGA-II) is explained by simulation in Section 5.2.3. The sequential

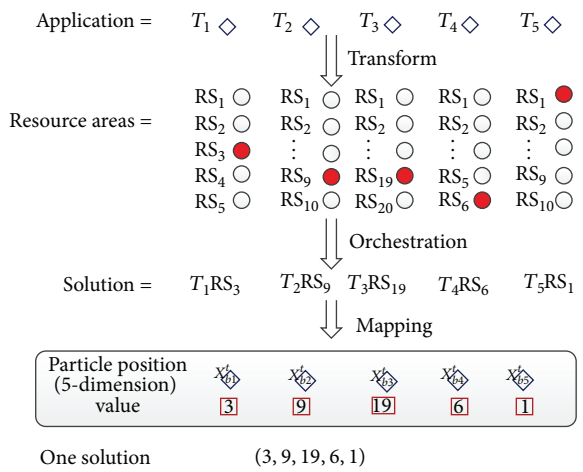


FIGURE 7: Mapping resource orchestration to MOPSO.

coding discrete PSO is adopted and the position of each particle represents a candidate solution to the problem. The number of tasks corresponds with the dimension numbers of the particle. Position values of a particle represent the resource service indexes for a task. The amount of Pareto optima may be even infinite since the conflicts of multiple objectives. Thus, multiobjective optimization by PSO differs from typical PSO algorithm in three aspects [36]. First, when a particle updates local optimum, if a new position (i.e., a new solution) and the local optimum are both nondominated solutions, we need to choose one solution to be the new local optimum. Second, when the swarm chooses one global optimum from all particles' local optima, a method is needed to choose one solution as the global optimum from nondominated local optima. Besides, nondominated local optima which are not chosen should be stored in an external archive to keep solutions' diversity. Third, there may be enormous nondominated local optima in all iterations. However, external archive can only store a certain amount of nondominated solutions. The update of external archive should keep nondominated solutions scattering evenly in the objective space.

4.3. Multiobjective PSO Algorithm. Assume that there are M tasks in the application. Then particle is set to M-dimension. Let $x_b^t = (x_{b1}^t, x_{b2}^t, \ldots, x_{bi}^t, \ldots, x_{bM}^t)$ be the particle's bth-dimensional position at the tth iteration and let $v_b^t = (v_{b1}^t, v_{b2}^t, \ldots, v_{bi}^t, \ldots, v_{bM}^t)$ be particle's velocity of bth-dimensional position at the tth iteration. Let $p_b^t = (p_{b1}^t, p_{b2}^t, \ldots, p_{bi}^t, \ldots, p_{bM}^t)$ and $p_g^t = (p_{g1}^t, p_{g2}^t, \ldots, p_{gi}^t, \ldots, p_{gM}^t)$ be the local and global optimum position, respectively, and let $A^t = \{p_g^t(1), \ldots, p_g^t(j), \ldots, p_g^t(O)\}$ be the external archive that contains O Pareto optima at most. $T_{\max}$ is the maximum iteration number. The algorithm is shown as follows:

/∗*Initialization Stage*∗/

(1) $t = 1$;

(2) initialize x_b^1 and v_b^1 randomly; $p_b^1 = x_b^1$;

(3) restrict positions and velocities according to the range of solution space;

(4) let temp$A^1 = \{p_g^1(1), \ldots, p_g^1(j), \ldots, p_g^1(Q^1)\}$ be Q^1 nondominated solutions from p_b^1 $(b = 1, \ldots, m)$;

(5) if $Q^1 \le O$, then $A^1 =$ tempA^1; else, select O nondominated solutions from tempA^1 to A^1 based on ascending crowding degrees;

(6) select a Pareto optimum as p_g^1 randomly from A^1.

/∗*Iteration Stage*∗/

(7) update particle's position and velocity with (13);

(8) restrict positions and velocities according to the range of solution space;

(9) t++;

(10) if $x_b^t \succ p_b^t$, then $p_b^t = x_b^t$; else if x_b^t and p_b^t are both nondominated, then randomly select p_b^t from x_b^t and p_b^t;

(11) let temp$A^t = \{p_g^t(1), \ldots, p_g^t(j), \ldots, p_g^t(Q^t)\}$ be Q^t nondominated solutions from p_b^t $(b = 1, \ldots, m)$ and A^t;

(12) if $Q^t \le O$, then $A^t =$ tempA^t; else, select O nondominated solutions from tempA^t to A^t based on ascending crowding degrees;

(13) select a Pareto optimum as p_g^t randomly from A^t;

(14) if $t \le T_{\max}$, go to (7);

(15) Output A^t.

4.3.1. Initialization Stage. The initialization stage is shown in lines (1)–(8). In line (2), the algorithm initializes particles scattering randomly in the solution space, and the local optima of particles are defined as their positions at this time. Next, positions and velocities are restricted according to the range of solution space in line (3). In lines (4)-(5), the external archive is initialized. At first, nondominated solutions in all local optima are selected to a temporary archive. If the amount of nondominated solutions is less than the capacity O of external archive, temporary archive will be assigned to the external archive. Otherwise, O nondominated solutions must be selected to be stored in the external archive. The selection is accomplished in three steps. First, objective vector of each nondominated solution in temporary archive is calculated. Second, the Euclidean distance of objective vectors is calculated to measuring the crowding degree of nondominated solutions. Third, O nondominated solutions with lower crowding degrees are selected to the external archive. In line (6), the global optimum is selected randomly from the external archive.

4.3.2. Iteration Stage. The iteration stage is shown in lines (7)–(14). The first step is the update of particles' positions and velocities. In the $(t+1)$th iteration, the bth particle's i-dimensional velocity is updated according to the velocity at last iteration, its local optimum p_{bi}^{t+1}, and global optimum p_{gi}^{t+1}. Then its i-dimensional position is updated by the updated velocity and the position at last iteration. This process is update by (9):

$$v_{bi}^{t+1} = \omega^t v_{bi}^t + c_1^t \xi^t \left(p_{bi}^t - x_{bi}^t\right) + c_2^t \eta^t \left(p_{gi}^t - x_{bi}^t\right),$$

$$x_{bi}^{t+1} = \begin{cases} \lceil x_{bi}^t + v_{bi}^{t+1} \rceil & \text{with prob. } p_1 = \left(x_{bi}^t + v_{bi}^{t+1}\right) - \lfloor x_{bi}^t + v_{bi}^{t+1} \rfloor \\ \lfloor x_{bi}^t + v_{bi}^{t+1} \rfloor & \text{with prob. } p_2 = 1 - p_1, \end{cases} \tag{13}$$

where ω^t is the time-variant inertia weight which controls the inheritance from the velocity at last iteration. c_1^t and c_2^t are time-variant learning factors which let the particle move towards local optimum and global optimum. ω^t, c_1^t, and c_2^t together with proper value can balance particles' exploration and exploitation capabilities. ξ^t and η^t are random numbers uniformly distributed between zero and one. They keep the particle moving randomly to escape suboptima. Since the value of x_{bi}^{t+1} represents the index of ith service instance, it must be an integer. If it is not an integer, it will be rounded to the nearest smaller integer or the nearest larger integer randomly such that the mean error is zero. Parameters in (13) are calculated as

$$\omega^t = \omega_{\max} - \left(\omega_{\max} - \omega_{\min}\right) \times \frac{t}{T_{\max}},$$

$$c_1^t = \left(c_{1f} - c_{1i}\right) \times \frac{t}{T_{\max}} + c_{1i}; \tag{14}$$

$$c_2^t = \left(c_{2f} - c_{2i}\right) \times \frac{t}{T_{\max}} + c_{2i},$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum value of inertia weight. $T_{\max}$ is the maximum iteration number. In (13), c_{1i} and c_{1f} are the initial and final values of c_1^t. c_{2i} and c_{2f} are the initial and final values of c_2^t.

Before the selection of local and global optimum of each particle, its velocity and position must be restricted according to the range of solution space in line (8). In lines (9)–(13), the update process of local optimum, external archive, and global optimum is quite similar to that in initialization stage.

Table 2: Simulation parameters.

MCC resource		Application		PSO	
E^r_{trans}	[0.01, 0.1]	V^i_{offload}	[10, 100]	$\omega_{\max}$ [21]	0.9
E_{comp}	0.05	E_{comp}	[1, 10]	$\omega_{\min}$ [21]	0.4
C^r_{comput}	[10, 50]	C^r_{comput}	[5, 30]	c_{1i} [22]	2.5
C^r_{commun}	[0.01, 0.1]	C^r_{commun}	[10, 50]	c_{1f} [22]	0.5
C^a_{trans}	[0.05, 0.5]			c_{2i} [22]	0.5
U_r	[0.05, 0.15]			c_{2f} [22]	2.5

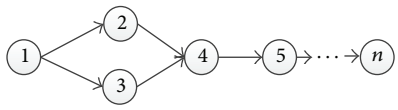


Figure 8: Application workflow in simulation scenarios.

The difference lies in the fact that nondominated solutions should be selected from local optima and external archive in line (13). Finally, if iteration number exceeds the maximum iteration limit, the algorithm outputs all Pareto optima in external archive.

5. Performance Evaluation

In this section, the performance of proposed resource orchestration system is evaluated. First, we evaluate how the parameters of mobility model affect the performance. Second, the visualized optimal solutions of resource selection with different size of candidate resources are presented, comparing with NSGA-II [37] which is a well-known multiobjective genetic algorithm. At last, the multiobjective metrics that can demonstrate the performance of our algorithm is analyzed, which shows that MOPSO is more suitable for resource orchestration problems than NSGA-II.

5.1. Simulation Scenarios. In the simulation scenarios, we generate various applications that include different number of tasks and have multiple MCC resource providers. For each component task, the volume of data V^i_{offload}, $V^i_{\text{interaction}}$, and V^{ij}_{commun} is generated according to a uniform distribution. Then we generate several resource providers to represent heterogeneous MCC resources with E^r_{trans}; C^r_{comput}, C^r_{commun}, and U_r follow a uniform distribution. The detailed setting is shown in Table 2. The resource providers are randomly classified into different resource sets and distributed to the tasks. We take an example of an application that contains sequential and parallel workflows depicted in Figure 8. The other structures can be transformed to the similar workflow by existing techniques [38].

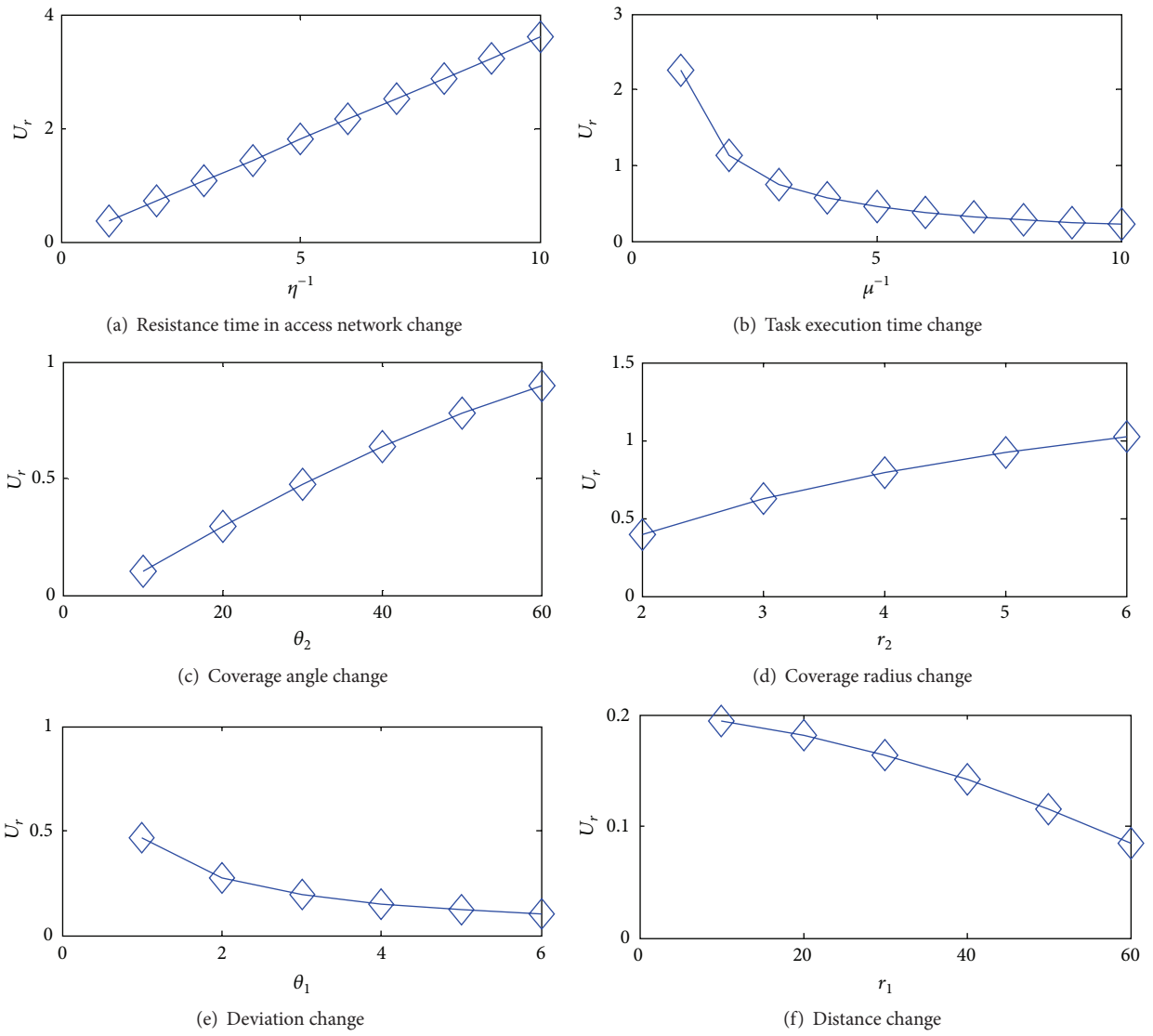
5.2. Simulation Results

5.2.1. Mobility Evaluation. We generate user's moving parameters according to the mobility model introduced in Section 4. Figure 9 depicts the evaluation results of the availability of infrastructure-based resources and mobile P2P resources. Figures 9(a) and 9(b) depict the effect of parameters resident time and task execution. When the resident time η^{-1} increases, the resource availability grows; but when the task execution time μ_i^{-1} increases, the resource availability declines. Therefore, if user keeps a high moving speed and the task execution time is long, the mobile device may suffer a network handoff, which affects the resource availability. From Figures 9(c) and 9(d), we can see that the availability value increases along with the coverage angle and coverage radius of mobile resource node. These results mean that the larger coverage area of mobile resource node leads to higher availability. In Figures 9(e) and 9(f), coverage areas of mobile node are not change. However, the deviation angle becomes bigger which means that the center of mobile resource node's location becomes further from the user's moving direction. In addition, the distance between user's mobile device and mobile resource node becomes bigger. Thus, from the above two cases, the availability values decrease.

5.2.2. Optimality Evaluation. To facilitate understanding results, we use the three-dimensional figures to show the optimal solutions in resource orchestration with energy consumption, cost, and availability as the objectives. The population sizes of MOPSO and NSGA-II are both 20. The external archive size of MOPSO is also 20. In each case, the iteration times of two algorithms are both 50. We compare two algorithms' performance in four cases. In case 1 and case 2, an application contains five and six tasks, respectively; and there are 10 resource candidates provided for each task. In case 3 and case 4, there are 20 resource candidates and 30 resource candidates provided for each task, respectively; and an application contains four tasks. The parameters are generated according to Table 2.

The true optimal and the Pareto fronts produced by MOPSO and NSGA-II are presented in Figure 10 with different task number and various resource area numbers. Circles represent the value of true Pareto optima in the objective space. These figures show that MOPSO can find more near-optimal Pareto front than NSGA-II. In addition, Pareto front of MOPSO covers wider true optima area than those of NSGA-II. However, the distribution of NSGA-II's Pareto front has a more uniform than that of MOPSO's Pareto front. Besides, when the number of tasks, that is, Figure 10(a) comparing with Figure 10(b), or the number of resources increases, Figure 10(c) comparing with Figure 10(d), the size true Pareto optima become large. Then the accuracies of all algorithms decrease with the number of resource candidates increases exponentially. Fortunately, the Resource Controller can be aware of the number of the resource candidates and adjust the particle number as well as optimal solution size, so that more choice can be provided to the users.

5.2.3. Multiobjective Evaluation. From the above analysis, we can see that the total size of resource candidate impacts the algorithm accuracy. The total size of resource candidate is the sum of MCC resource numbers provided for all the tasks included in an application. Here, we compare algorithm metrics with total size of resource candidates to analyze the



(a) Resistance time in access network change

(b) Task execution time change

(c) Coverage angle change

(d) Coverage radius change

(e) Deviation change

(f) Distance change

Figure 9: Availability of MCC resources with different mobility parameters.

effective multiobjective algorithm and verify the satisfaction degree of the algorithm. Zitzler et al. [39] have proved that at least M metrics are needed to compare two algorithms for an optimization problem with M objectives. Therefore, we use generational distance (GD), spacing (S), and maximum spread (MS) for algorithms' performance evaluation. These metrics correspond to algorithms' approximation, uniformity, and coverage respectively. Besides, we use execution time for evaluating algorithms' real-time capabilities.

GD is used to calculate the distance of true Pareto front $\mathrm{PF}_{\mathrm{true}}$ and Pareto front $\mathrm{PF}_{\mathrm{known}}$ found by the algorithm [40], which can be calculated by

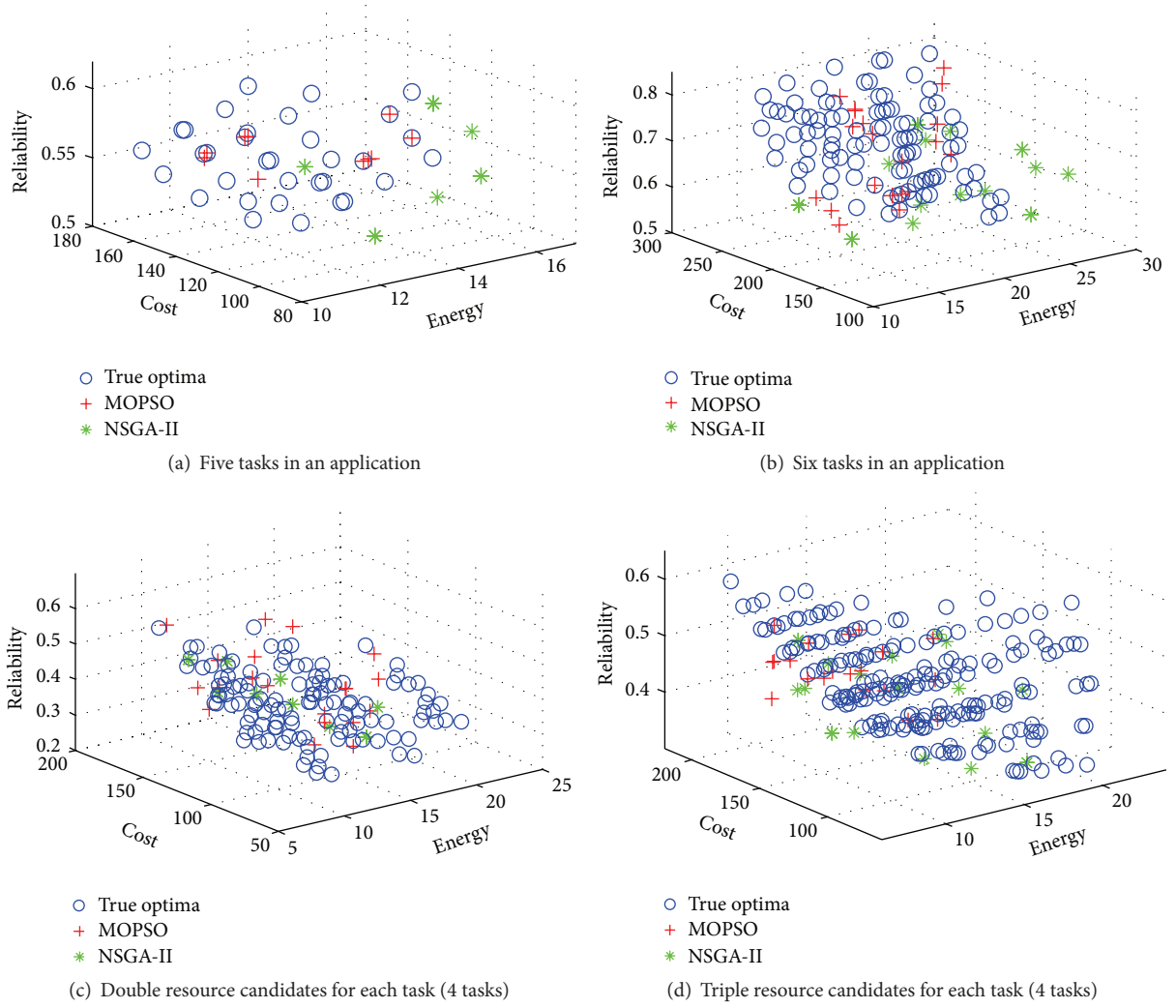
$$\mathrm{GD} = \left(\frac{1}{n_{\mathrm{PF}}} \sum_{i=1}^{n_{\mathrm{PF}}} d_i^{\,2} \right)^{1/2}, \tag{15}$$

where n_{PF} is the solution amount of $\mathrm{PF}_{\mathrm{known}}$. d_i is the Euclidean distance of ith solution in $\mathrm{PF}_{\mathrm{known}}$ and the closest solution in $\mathrm{PF}_{\mathrm{true}}$. If $\mathrm{GD} = 0$, $\mathrm{PF}_{\mathrm{known}} = \mathrm{PF}_{\mathrm{true}}$. Otherwise, GD indicates the extent to which $\mathrm{PF}_{\mathrm{known}}$ deviates from $\mathrm{PF}_{\mathrm{true}}$.

S is the space metric used to measure distribution uniformity of solutions in $\mathrm{PF}_{\mathrm{known}}$ [41], which can be calculated by

$$S = \frac{\left[(1/n_{\mathrm{PF}}) \sum_{i=1}^{n_{\mathrm{PF}}} \left(d_i' - \overline{d}' \right)^2 \right]^{1/2}}{\overline{d}'}, \tag{16}$$

where $\overline{d}' = (1/n_{\mathrm{PF}}) \sum_{i=1}^{n_{\mathrm{PF}}} d_i'$. n_{PF} is the solution amount of $\mathrm{PF}_{\mathrm{known}}$. d_i' is the Euclidean distance of ith solution in $\mathrm{PF}_{\mathrm{known}}$ and the closest solution in $\mathrm{PF}_{\mathrm{known}}$. If $S = 0$, solutions in $\mathrm{PF}_{\mathrm{known}}$ are distributed evenly. Otherwise, S indicates the nonuniformity of $\mathrm{PF}_{\mathrm{known}}$.



(a) Five tasks in an application

(b) Six tasks in an application

(c) Double resource candidates for each task (4 tasks)

(d) Triple resource candidates for each task (4 tasks)

Figure 10: Comparisons of resource orchestration results.

MS is used to measure the extent to which PF_{known} covers PF_{true} by using hyper-boxes formed by objective extrema in PF_{known} and PF_{true} [42], which can be calculated by

$$MS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{f_i^{max} - f_i^{min}}{F_i^{max} - F_i^{min}}\right)^2}, \quad (17)$$

where n is the objective number. f_i^{max} and f_i^{min} are maximum and minimum value of ith objective in PF_{known} respectively. F_i^{max} and F_i^{min} are maximum and minimum value of ith objective in PF_{true}, respectively. Larger MS means better coverage performance.

The box plots of GD, S, MS, and execution time of MOPSO and NSGA-II for the MCC resource orchestration are presented in Figures 11, 12, and 13. For each experiment, each algorithm is executed 50 times to draw concrete results. In these figures, the total size of resource candidates is 10^3, 10^4, and 10^5, respectively. In Figure 11(a), GD value of MOPSO mainly spreads below 16, which is smaller than that of NSGA-II. It illustrates that the Pareto front found by MOPSO is closer to the true Pareto front than that found by NSGA-II; that is, MOPSO is better than NSGA-II in approximation. As shown in Figure 11(b), S value of NSGA-II is much smaller than that of MOPSO. The Pareto front found by NSGA-II scatters more evenly than that found by MOPSO. This weakness may result in the premature convergence of proposed algorithm. The distributions of MS value produced by two algorithms are very similar in Figure 11(c). MS of MOPSO outperforms that of NSGA-II slightly, which means that the Pareto front of MOPSO covers more areas in the solution space. The execution time of MOPSO is quite smaller than that of NSGA-II as shown in Figure 11(d). This advantage is extremely significant when massive concurrent service requests arrive at the system. Hence, MOPSO is more suitable for the resource orchestration of real-time applications.

With the exponential increase of resource candidates in Figures 12 and 13, GD value increases accordingly, which means that accuracies of these two algorithms are worsening. In this process, GD value of MOPSO is still smaller than that of NSGA-II; that is, the approximation of MOPSO surpasses that of NSGA-II. Although S value of NSGA-II is below

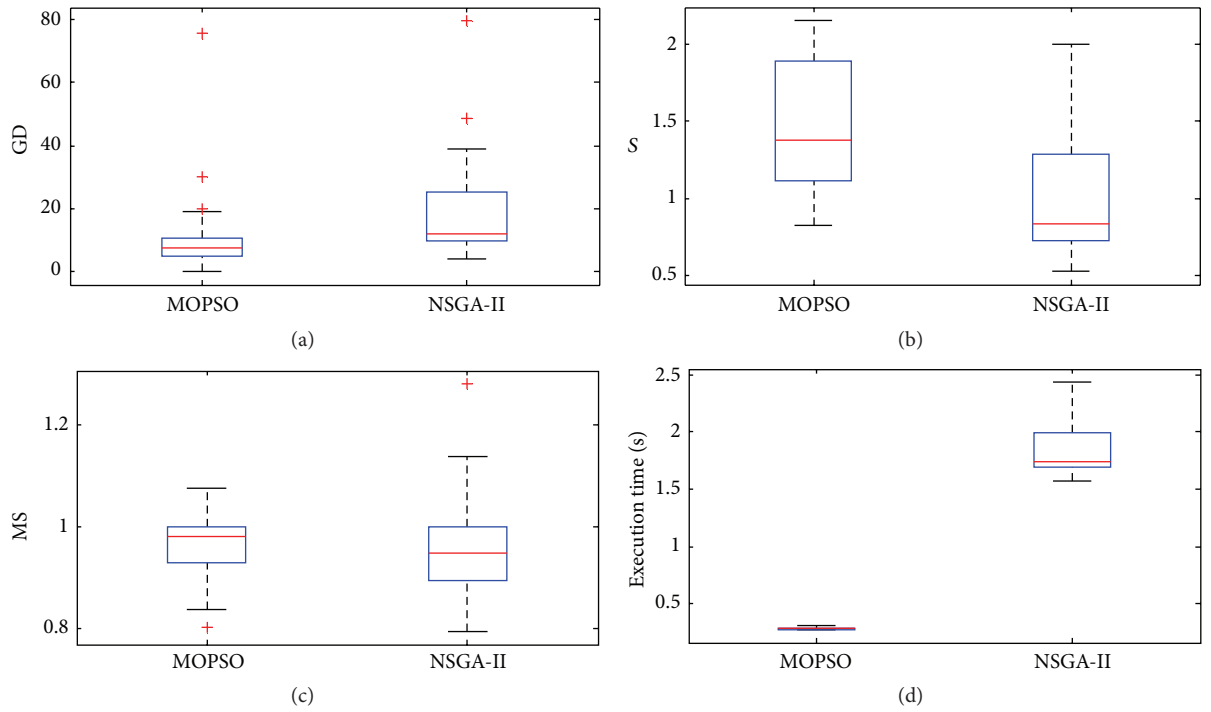


Figure 11: GD, S, MS, and execution time for resource orchestration when task candidate number is 10^3.

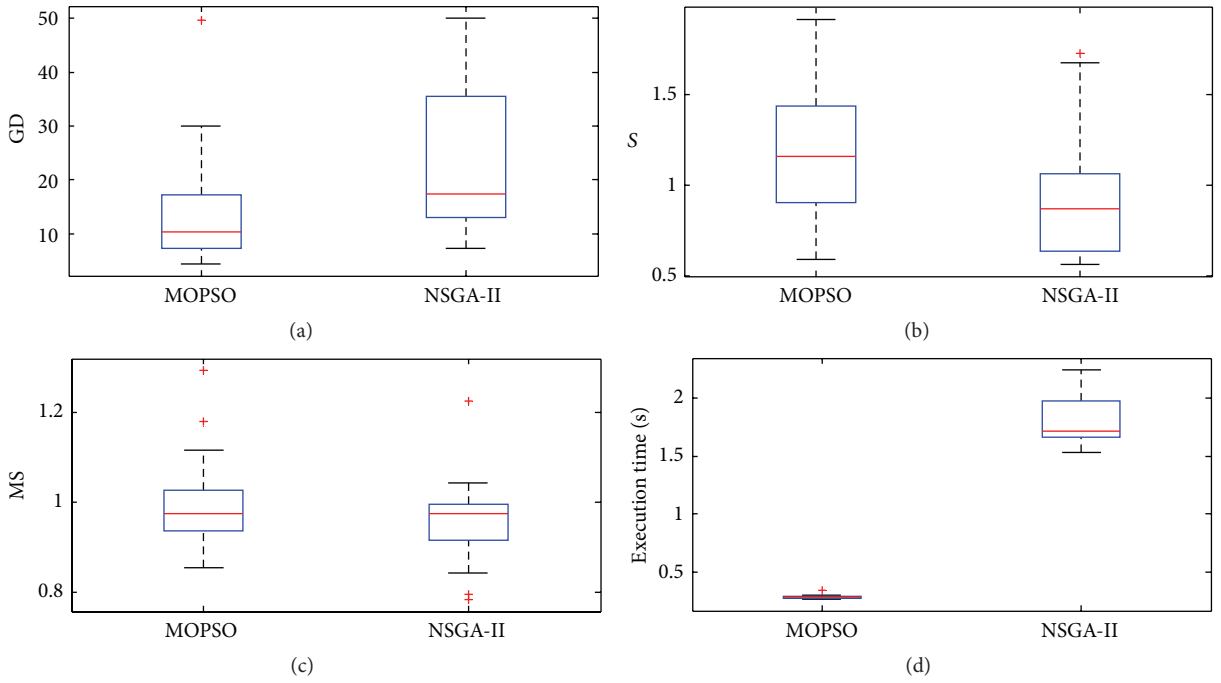


Figure 12: GD, S, MS, and execution time for resource orchestration when task candidate number is 10^4.

that of MOPSO in this process, the uniformity of MOPSO approaches that of NSGA-II. MS value and execution time of these two algorithms in Figures 12 and 13 are similar to those when the total size of resource candidates is 10^3.

To conclude simulation results, MOPSO surpass NSGA-II enormously in approximation and execution time. The advantage of MOPSO is slight in coverage. Although MOPSO is worse than NSGA-II in uniformity, this disadvantage diminishes with the increase of solution space. These simulation results demonstrate that the proposed algorithm is more effective and efficient than the comparative algorithm in real-time resource orchestration situations.

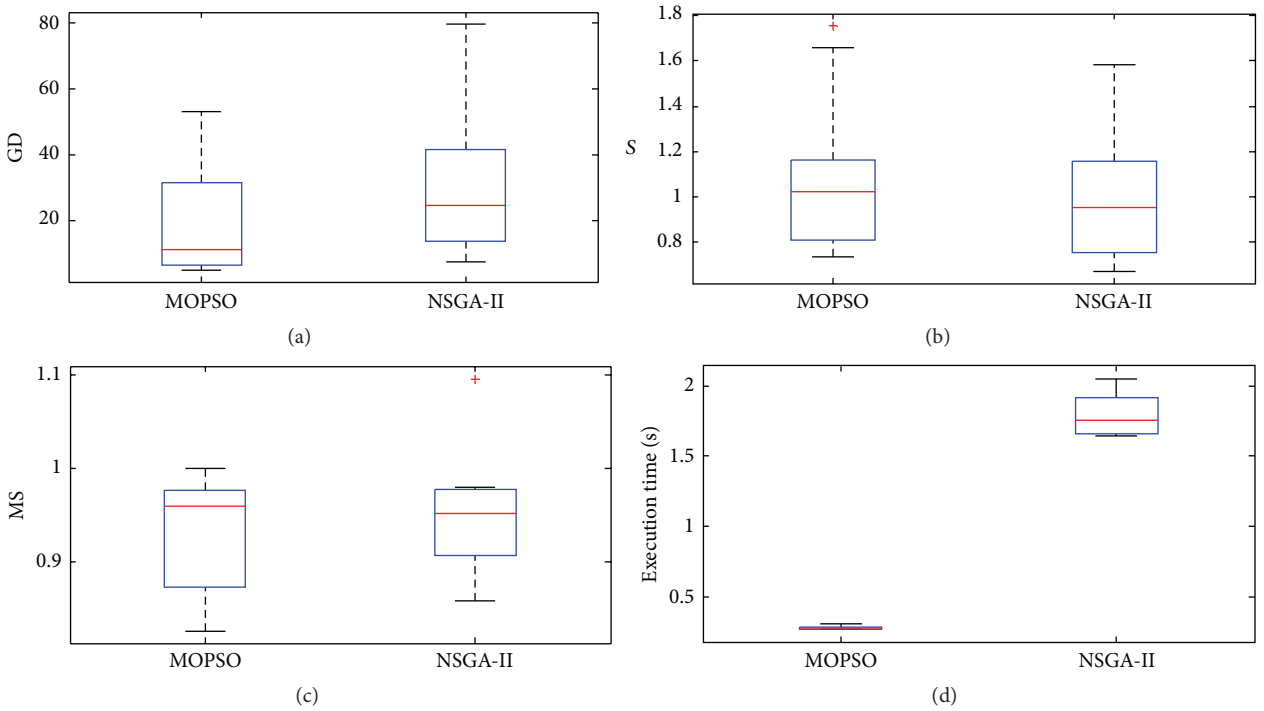


Figure 13: GD, S, MS, and execution time for resource orchestration when task candidate number is 10^5.

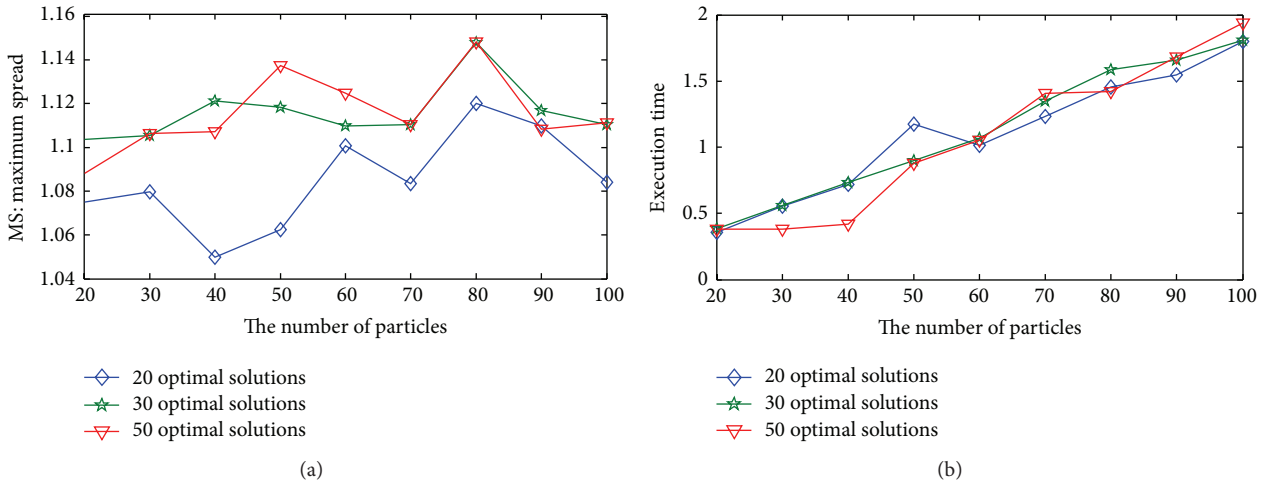


Figure 14: The comparison of convergence and execution time.

Finally, the performance of the multiobjective algorithms in terms of convergence time is analyzed. From Figure 14, taking more particles may increase the MS, which means better coverage performance, but it also results in the increasing of execution time. On the other hand, we can see that setting large optimal size, that is, from 20 solutions to 50 solutions, does not affect the convergence. Therefore, more particles with longer execution time can make the better convergence. Sometimes it is acceptable to have solutions of lower quality that can be computed fast rather than optimal solutions obtained after long time computations. The resource orchestration system can provide these choices for users.

6. Conclusion

This paper targets the problem of resource orchestration in MCC environment. Our work goes beyond existing approaches by considering resource choice for complicated application where multiple tasks are composed together in a specific business process, while others mainly focus on single

service or single offload destinations. By taking use of SDN idea to decouple mobile resource control from all user plane elements, the software defined Resource Controller is proposed for making the orchestration strategies and realizing resource allocation. Based on the software defined system, we formulate the resource orchestration as optimization problem with multiobjective, that is, energy consumption of mobile device, resource charge, and resource availability that contains different mobility model for the three types of MCC resources. The multiobjective PSO algorithm is used to solve the resource orchestration problem and obtains the approximate optimal solutions in acceptable time. Simulation results show that the multiple solutions for all of our studied cases almost can archive part of the Pareto optimum and are more effective and efficient than NSGA-II algorithm in approximation, coverage, and execution time. Our work goes beyond existing approaches by considering resource choice for complicated application where multiple tasks are composed together in a specific business process, while others mainly focus on single service or single offload destinations.

The resource orchestration in MCC is suitable for the application offload in IaaS, PaaS, SaaS, and the cross-layer resource management. The multiobjective metrics can be changed according to the requirement of application. In addition, MCC resource orchestration system can support the service placing of Network Functions Virtualization (NFV) in mobile network, which solves its selecting problem of deployment resource.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was jointly supported by (1) the National Basic Research Program of China (no. 2013CB329102); (2) National Natural Science Foundation of China (nos. 61471063, 61421061, 61372120, 61302087, and 61271019); (3) the Key (Keygrant) Project of Chinese Ministry of Education (no. MCM20130310); (4) Beijing Higher Education Young Elite Teacher Project (no. YETP0473).

References

[1] W. Li, Y. Zhao, S. Lu, and D. Chen, "Mechanisms and challenges on mobility-augmented service provisioning for mobile cloud computing," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 89–97, 2015.

[2] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[3] M. Satyanarayanan, R. Schuster, M. Ebling et al., "An open ecosystem for mobile-cloud convergence," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 63–70, 2015.

[4] M. Fazio and A. Puliafito, "Cloud4sens: a cloud-based architecture for sensor controlling and monitoring," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 41–47, 2015.

[5] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *IEEE Network*, vol. 27, no. 5, pp. 34–40, 2013.

[6] A. Manzalini and N. Crespi, "An edge operating system enabling anything-as-a-service," *IEEE Communications Magazine*, vol. 54, no. 3, pp. 62–67, 2016.

[7] J. Lee, Y. Turner, M. Lee et al., "Application-driven bandwidth guarantees in datacenters," in *Proceedings of the ACM Conference on Special Interest Group on Data Communication (SIGCOMM '14)*, pp. 467–478, Chicago, Ill, USA, August 2014.

[8] R. Ranjan, B. Benatallah, S. Dustdar, and M. P. Papazoglou, "Cloud resource orchestration programming: overview, issues, and directions," *IEEE Internet Computing*, vol. 19, no. 5, pp. 46–56, 2015.

[9] S. Mustafa, B. Nazir, A. Hayat, A. U. R. Khan, and S. A. Madani, "Resource management in cloud computing: taxonomy, prospects, and challenges," *Computers & Electrical Engineering*, vol. 47, pp. 186–203, 2015.

[10] Y. Ge, Y. Zhang, Q. Qiu, and Y. H. Lu, "A game theoretic resource allocation for overall energy minimization in mobile cloud computing system," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '12)*, Redondo Beach, Calif, USA, July-August 2012.

[11] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Proceedings of the IEEE 7th International Symposium on Service-Oriented System Engineering (SOSE '13)*, pp. 494–502, Redwood City, Calif, USA, March 2013.

[12] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2685–2700, 2013.

[13] V. Misra, S. Ioannidis, A. Chaintreau, and L. Massoulié, "Incentivizing peer-assisted services: a fluid shapley value approach," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '10)*, pp. 215–226, New York, NY, USA, June 2010.

[14] R. Gabner, H.-P. Schwefel, K. A. Hummel, and G. Haring, "Optimal model-based policies for component migration of mobile cloud services," in *Proceedings of the 10th IEEE International Symposium on Network Computing and Applications (NCA '11)*, pp. 195–202, IEEE, Cambridge, Mass, USA, August 2011.

[15] H. Liang, L. X. Cai, D. Huang, X. Shen, and D. Peng, "An SMDP-based service model for interdomain resource allocation in mobile cloud networks," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 5, pp. 2222–2232, 2012.

[16] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3317–3329, 2015.

[17] T. Verbelen, T. Stevens, F. De Turck, and B. Dhoedt, "Graph partitioning algorithms for optimizing software deployment in mobile cloud computing," *Future Generation Computer Systems*, vol. 29, no. 2, pp. 451–459, 2013.

[18] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review Archive*, vol. 40, no. 4, pp. 23–32, 2013.

[19] L. F. Bittencourt, E. R. M. Madeira, and N. L. S. Da Fonseca, "Scheduling in hybrid clouds," *IEEE Communications Magazine*, vol. 50, no. 9, pp. 42–47, 2012.

[20] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, 2014.

[21] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, pp. 1945–1950, Washington, DC, USA, July 1999.

[22] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[23] C. Li and L. Li, "Energy constrained resource allocation optimization for mobile grids," *Journal of Parallel and Distributed Computing*, vol. 70, no. 3, pp. 245–258, 2010.

[24] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: from concept to practice and beyond," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 80–88, 2015.

[25] M. Bienkowski, A. Feldmann, J. Grassler, G. Schaffrath, and S. Schmid, "The wide-area virtual service migration problem: a competitive analysis approach," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 165–178, 2014.

[26] X. Gu, K. Nahrstedt, A. Messer, I. Greenberg, and D. Milojicic, "Adaptive offloading for pervasive computing," *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 66–73, 2004.

[27] "Leadership in enabling and industrial technologies: Information and Communication Technologies," Horizon 2020 Work Programme, 2014-2015.

[28] G. Breiter, M. Behrendt, M. Gupta et al., "Software defined environments based on TOSCA in IBM cloud implementations," *IBM Journal of Research and Development*, vol. 58, no. 2, Article ID 6798738, 2014.

[29] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1294–1313, 2013.

[30] T. Taleb and A. Ksentini, "Follow Me cloud: interworking federated clouds and distributed mobile networks," *IEEE Network*, vol. 27, no. 5, pp. 12–19, 2013.

[31] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 2716–2720, Orlando, Fla, USA, March 2012.

[32] Y.-B. Lin, "Reducing location update cost in a PCS network," *IEEE/ACM Transactions on Networking*, vol. 5, no. 1, pp. 25–33, 1997.

[33] M. C. González, C. A. Hidalgo, and A.-L. Barabási, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[34] S.-Y. Wu, J. Hsu, and C.-M. Chen, "Headlight prefetching and dynamic chaining for cooperative media streaming in mobile environments," *IEEE Transactions on Mobile Computing*, vol. 8, no. 2, pp. 173–187, 2009.

[35] F. Tao, D. Zhao, Y. Hu, and Z. Zhou, "Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 4, pp. 315–327, 2008.

[36] J. Liao, Y. Liu, X. Zhu, J. Wang, and Q. Qi, "A multi-objective service selection algorithm for service composition," in *Proceedings of the 19th Asia-Pacific Conference on Communications (APCC '13)*, pp. 75–80, Denpasar, Indonesia, August 2013.

[37] 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.

[38] J. Liao, Y. Liu, X. Zhu, and J. Wang, "Accurate sub-swarms particle swarm optimization algorithm for service composition," *Journal of Systems and Software*, vol. 90, no. 1, pp. 191–203, 2014.

[39] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[40] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, pp. 204–211, La Jolla, Calif, USA, July 2000.

[41] J. R. Schott, *Fault tolerant design using single and multicriteria genetic algorithm optimization [M.S. thesis]*, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1995.

[42] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.