Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches

ZHI-HUI ZHAN, XIAO-FANG LIU, YUE-JIAO GONG, and JUN ZHANG,
Sun Yat-sen University
HENRY SHU-HUNG CHUNG, City University of Hong Kong
YUN LI, University of Glasgow

A disruptive technology fundamentally transforming the way that computing services are delivered, cloud computing offers information and communication technology users a new dimension of convenience of resources, as services via the Internet. Because cloud provides a finite pool of virtualized on-demand resources, optimally scheduling them has become an essential and rewarding topic, where a trend of using Evolutionary Computation (EC) algorithms is emerging rapidly. Through analyzing the cloud computing architecture, this survey first presents taxonomy at two levels of scheduling cloud resources. It then paints a landscape of the scheduling problem and solutions. According to the taxonomy, a comprehensive survey of state-of-the-art approaches is presented systematically. Looking forward, challenges and potential future research directions are investigated and invited, including real-time scheduling, adaptive dynamic scheduling, large-scale scheduling, multiobjective scheduling, and distributed and parallel scheduling. At the dawn of Industry 4.0, cloud computing scheduling for cyber-physical integration with the presence of big data is also discussed. Research in this area is only in its infancy, but with the rapid fusion of information and data technology, more exciting and agenda-setting topics are likely to emerge on the horizon.

Categories and Subject Descriptors: I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization*; D.4.1 [**Process Management**]: Scheduling; H.3.5 [**Online Information Services**]: Web-based Services; Commercial Services; H.3.4 [**Systems and Software**]: Distributed Systems; Performance Evaluation; C.1.4 [**Parallel Architectures**]: Distributed Architectures; C.2.1 [**Network Architecture and Design**]: Network Communications; Network Topology; Wireless Communication; C.2.3 [**Network Operations**]: Network Management; Network Monitoring; C.5.5 [**Servers**]; K.1 [**The Computer Industry**]: Suppliers

General Terms: Algorithms, Management, Design, Performance

Additional Key Words and Phrases: Cloud computing, resource scheduling, evolutionary computation, genetic algorithm, ant colony optimization, particle swarm optimization

---

This work was supported in part by the National Natural Science Foundations of China (NSFC) with No. 61402545, the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars with No. 2014A030306038, the Project for Pearl River New Star in Science and Technology with No. 201506010047, the Fundamental Research Funds for the Central Universities, the NSFC Key Program with No. 61332002, the NSFC for Distinguished Young Scholars with No. 61125205, and the National High-Technology Research and Development Program (863 Program) of China with No. 2013AA01A212.
Authors' addresses: Z.-H. Zhan (corresponding author), X.-F. Liu, Y.-J. Gong, and J. Zhang, School of Advanced Computing, Sun Yat-sen University, Guangzhou 510275, China, with the Key Laboratory of Machine Intelligence and Advanced Computing, Sun Yat-sen University, Ministry of Education, China, with the Engineering Research Center of Supercomputing Engineering Software, Sun Yat-sen University, Ministry of Education, China, and with the Key Laboratory of Software Technology, Sun Yat-sen University, Education Department of Guangdong Province, China; email: zhanzhh@mail.sysu.edu.cn; H. S.-H. Chung, Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong, SAR, China; Y. Li, School of Engineering, University of Glasgow, Glasgow G12 8LT, UK.

ACM 0360-0300/2015/07-ART63 $15.00
DOI: http://dx.doi.org/10.1145/2788397

ACM Reference Format:
Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. 2015. Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput. Surv. 47, 4, Article 63 (July 2015), 33 pages.
DOI: http://dx.doi.org/10.1145/2788397

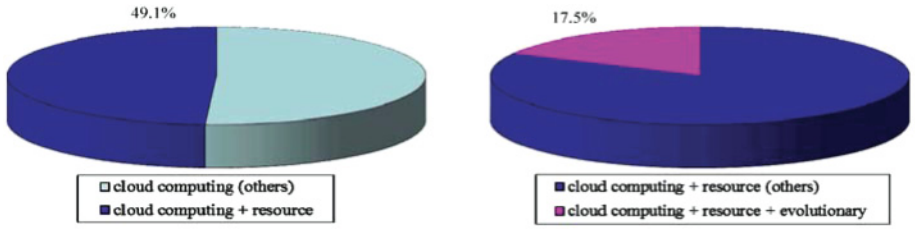
1. INTRODUCTION

With the ubiquitous growth of Internet access and of big data in their volume, velocity, and variety through the Internet, cloud computing becomes more and more proliferating in industry, academia, and society [Dragland 2013; Toosi et al. 2014]. The "cloud" provides worldwide users, being either stationary or on the move, with a new dimension of computing resources in the form of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [Zhang et al. 2010]. These types of resources are provided on demand in a pay-per-use/subscription fashion in an Internet-based environment [Foster et al. 2008; Xu et al. 2014].

Consisting of a large number of resources as a federation, cloud computing shares some characteristics common to parallel computing such as cluster computing and grid computing, but differs in that it uses virtualization for resources management [Grandison et al. 2010]. This allows computing resources to be scheduled for delivery to the end-user as a utility, dubbed the fifth utility, in a similar way to traditional utilities, namely, water, electricity, natural gas, and telephony [Buyya et al. 2009]. For example, massively elastic computing nodes are now disposable at one's fingertips, anywhere and anytime, with payment only required when it is used [Toosi et al. 2014]. This would not have been achievable with a traditional computing means or data center.

The explosive demand in cloud computing poses a challenge to, as well as opens up, a new area of research, where the traditional boundary of computing resources is blurred by virtualization. As cloud computing is a market-oriented utility, optimally scheduling resources can allow the cloud providers and users separately to focus on their own businesses to maximize their profit and return on investment [Armbrust et al. 2010]. In meeting Service Level Agreements (SLAs), for example, optimally scheduling resources is seen as of paramount importance to cloud computing services [Morshedlou and Meybodi 2014]. Underestimating the provision of resources would lead to broken SLAs and penalties, while overestimating the provision would lead to resource underutilization and lost revenue [Dikaiakos et al. 2009]. Cloud resource scheduling is also a challenging problem for the provider's resource-economic and power-efficient requirements. Further, it is a significant issue to the user in terms of Quality of Service (QoS) requirements under pay-per-use provision [Baliga et al. 2011].

Much work tackling the cloud resource scheduling problem has been reported in the literature, although mostly in conference proceedings due to the dynamically emerging nature of the problem [Heilig and Vob 2014; Toosi et al. 2014]. Google Scholar, for example, has reported 195,000 entries under the keyword of "cloud computing" and 95,800 entries under "cloud computing+resource" from 2009 to 2014, as depicted in Figure 1(a). This helps one visualize the significant role that resource management and scheduling play in cloud computing. In both academia and industry, the problem of cloud resource scheduling is seen to be as hard as a Nondeterministic Polynomial (NP) optimization problem [Genez et al. 2012; Xu et al. 2013], that is, an NP-hard problem, whose intractability increases exponentially with the number of variables if using a deterministic algorithm such as exhaustive search is attempted. Hence, algorithms that solve relatively routine cloud scheduling problems can suffer from a dimensionality breakdown when the size of the problem grows. This problem becomes



(a) Resource related to cloud computing (b) Evolutionary related to cloud computing resource

Fig. 1. Shares of cloud computing activities during 2009–2014, according to Google Scholar.

more challenging with the increase of the proliferation, ambition, and complexity of cloud computing.

Using an Evolutionary Computation (EC) algorithm to tackle cloud resource scheduling has received increasing attention in recent years, as such an algorithm offers an NP-hard problem global solution acceptable in a time frame proportional to the number of variables [Back et al. 2008; Zhang et al. 2011c; Li et al. 2014; Chen et al. 2014; Li et al. 2015c]. Following their successes in resource scheduling for grid computing [Yu et al. 2008], EC algorithms have gained momentum in tackling the increasing structural complexity of cloud resource management and scheduling [Jennings and Stadler 2014; Rodriguez and Buyya 2014]. As reported in Google Scholar and summarized in Figure 1(b), over 17.5% of the works involving "resources" are estimated concerning the use of an evolutionary method.

Although there exist surveys on "green cloud computing" [Baliga et al. 2011], cloud workflow scheduling [Bardsiri and Hashemi 2012], cloud task scheduling [Kaur and Singh 2012; Chawla and Bhonsle 2012], virtual machine performance management in cloud computing [Xu et al. 2014], and "interconnected cloud computing" [Toosi et al. 2014], they have not discussed EC algorithms in solving these problems. This article aims to provide an overview of the problem and a systematic review of the state-of-the-art techniques for cloud resource scheduling, with contributions including

—a two-leveled taxonomic structure, through analysis of the scheduling layers;
—the resultant clarity of landscape of the cloud scheduling problem;
—an ensuing systematic survey on the state-of-the-art approaches to cloud scheduling with analysis of their pros and cons;
—suggestions on how to use various EC approaches to various layers of cloud resource scheduling;
—analysis of challenges ahead and potential future research directions, such as real-time, adaptive dynamic, large-scale, multiobjective, and distributed scheduling; and
—outlook at the dawn of Industry 4.0, with the rapid fusion of information and data technology for cloud-based cyber-physical integration.

The rest of the article is organized as follows. A framework of the problem is presented in Section 2, summarized with a two-leveled taxonomic structure. This helps classify the rapidly advancing research and, according to the taxonomy, present survey details in Sections 3, 4, and 5 systematically. Publications surveyed were selected by considering their quality, popularity, number of citations, and coverage of different aspects of the topic. In Section 6, challenges and future directions of research are discussed and highlighted. These are followed by a concluding summary in Section 7.

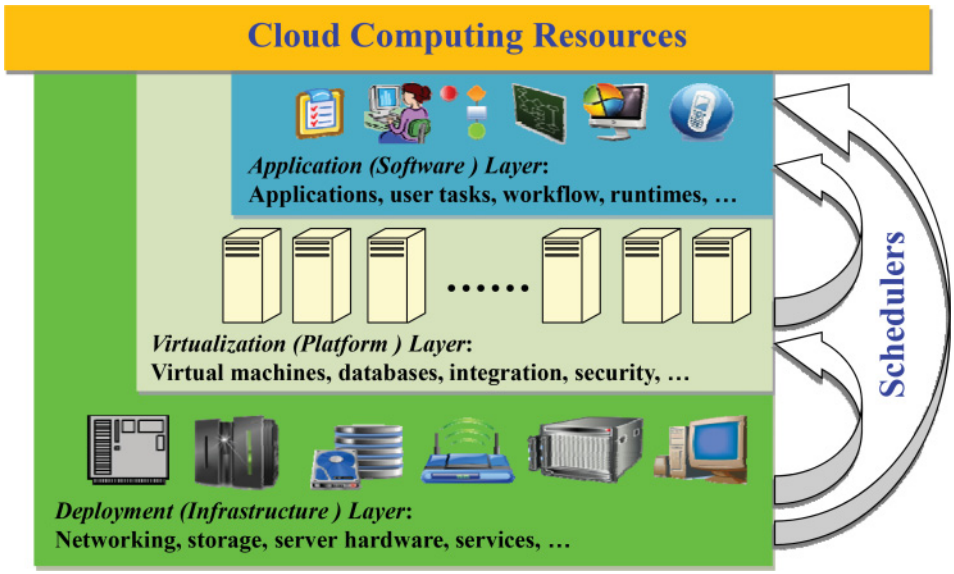


Fig. 2. Scheduling cloud computing resources at their service stacks.

2. CLOUD RESOURCE SCHEDULING

2.1. Cloud Resource Layers and Scheduling Taxonomy

When managing resources in a cloud computing environment, scheduling can be made in different layers of the service stacks. Hence, the architecture consisting of the IaaS, PaaS, and SaaS stacks can be used to classify cloud scheduling problems into"*scheduling in the application (software) layer*," "*scheduling in the virtualization (platform) layer*," and "*scheduling in the deployment (infrastructure) layer*," as illustrated in Figure 2 and explained next.

—*Scheduling in the application layer* is to schedule the virtual or physical resources to support software and user applications, tasks, and workflows, etc., with optimal QoS and efficiency.
—*Scheduling in the virtualization layer* focuses on mapping virtual resources onto physical resources with optimal load balance, energy conservation, etc.
—*Scheduling in the deployment layer*, which also attracts worldwide attention, is concerned with optimal and strategic infrastructure, outsourcing, service placement, multicloud centers, partnering, data routing and application migration, etc. [Toosi et al. 2014].

The preceding three categories form a high-level framework and taxonomy of the cloud resource scheduling problem. Low-level taxonomy can arrive from different scheduling objectives. First, when scheduling resources in the application layer, challenges do not only come from deadline and budget constraints of the cloud user, but also come from the cloud provider in that the resources need to be utilized with balance or a maximal rate. Therefore, this category is further divided into the subcategories of

—*"scheduling for user QoS,"*
—*"scheduling for provider efficiency,"* and
—*"scheduling for negotiation."*

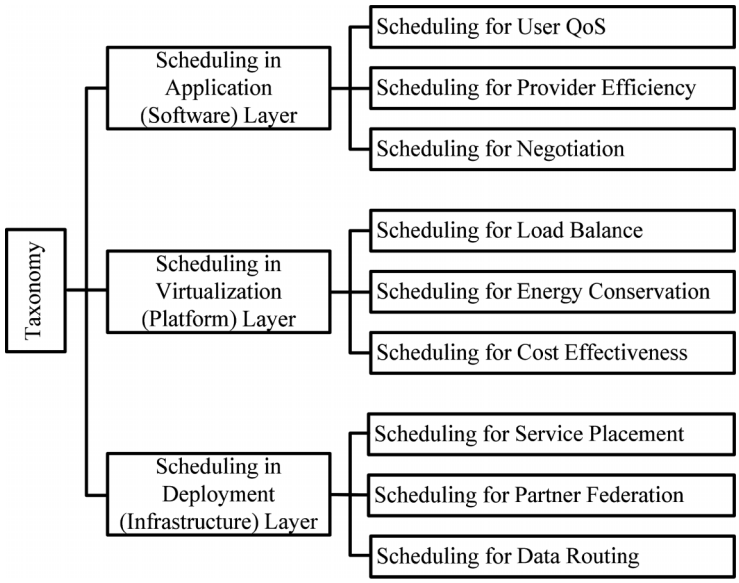


Fig. 3. Taxonomy of the cloud resource scheduling: The higher-level taxonomy is classified from the service stack architecture and the lower-level taxonomy from different scheduling challenges and objectives.

Secondly, when scheduling in the virtualization layer, challenges include how to schedule Virtual Machines (VMs) onto Physical Machines (PMs) efficiently with load balance, energy conservation, and/or cost effectiveness. Therefore, this category is divided into the subcategories of

—*"scheduling for load balance,"*
—*"scheduling for energy conservation,"* and
—*'scheduling for cost effectiveness."*

Finally, the deployment category is divided into the subcategories of

—*"scheduling for service placement,"*
—*"scheduling for partner federation,"* and
—*"scheduling for data routing."*

A chart of the taxonomy is thus completed in Figure 3.

2.2. Cloud Resource Scheduling Problem

Similar to an ordinary scheduling problem, cloud resource scheduling is to find an "optimal" mapping $C : \mathbf{T} \times \mathbf{R} \rightarrow \Re^{\mathbf{z}}$ that assigns M required tasks (or virtual resources) $\mathbf{T} = \{T_1, T_2, \ldots, T_M\}$ onto N available cloud (or physical) resources $\mathbf{R} = \{R_1, R_2, \ldots, R_N\}$ such that the fitness of z given objectives $\mathbf{F} = \{F_1, F_2, \ldots, F_z\}$ are maximized either collectively in weighting or individually in a Pareto sense within a given time frame. To help with the description, a list of nomenclature used in this survey is summarized in Table I.

This definition of the Cloud Resource Scheduling Problem is illustrated in Figure 4, where, for example, two tasks may have to share one resource. The objectives often include those falling into those categories listed in the low-level taxonomy, such as QoS, costs, energy conservation, load balance, task migration, bandwidth balance, utilization, reliabilities, scalability, etc., as well as negotiations.

Table I. Nomenclature Used in the Survey

Symbol	Description
M	The number of tasks to be scheduled on cloud resources The number of virtual resources to be scheduled on the physical resources
N	The number of cloud resources that execute the cloud tasks The number of physical resources that host the virtual resources
T_i	The i^{th} item (task) to be scheduled
R_j	The j^{th} cloud resource
u	The population size of candidate solutions
G	The maximum number of generations to run

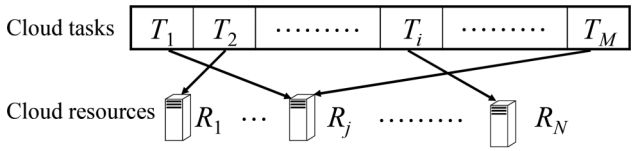


Fig. 4. A cloud resource scheduling problem: Mapping M cloud tasks onto N resources.

2.3. Cloud Resource Scheduling Algorithms

There exist scheduling algorithms that solve relatively routine problems in cloud computing, as reported in Bardsiri and Hashemi [2012], Kaur and Singh [2012], and Chawla and Bhonsle [2012]. These algorithms are often exhaustive in nature and can hence work well if the size of the scheduling problem is manageable through converting it to a combinatorial optimization problem such as a Linear Programming (LP) [Kumar and Balasubramanie 2012; Speitkamp and Bichler 2010], Integer Programming (IP) [Li and Guo 2010], Integer Linear Programming (ILP) [Genez et al. 2012], or constraint satisfaction problem [Van et al. 2010]. However, being an NP-hard problem, cloud scheduling can break down exhaustive or enumerative approaches with an increase in dimensionality or in the number of variables to be optimized.

Therefore, a heuristic method, such as that based on LP relaxation, has been adopted by Speitkamp and Bichler [2010] for minimizing server costs when scheduling VMs onto PMs. Also, two ILP-relaxation-based heuristics have been used by Genez et al. [2012] to schedule cloud resources so as to execute user tasks with two SLA levels from both the cloud user and the provider. Li and Guo [2010] have applied a minimized geometric Buchberger algorithm to a stochastic IP problem for SLA-aware resource scheduling. Xu et al. [2013] have also proposed heuristic methods to solve the problem of scheduling VMs on PMs. A heuristic backtracking algorithm, which performs a "depth first" search-like method to enumerate a subset of possible VM placement solutions, and a LP rounding based optimal algorithm for virtual link mapping are proposed to determine how to map user requirements to a data center network with minimum operational costs.

Most reported heuristic approaches are single-point based. Although single-point heuristic approaches and a greedy approach like First-Fit Decreasing (FFD) [Ajiro and Tanaka 2007] can be fast to find a nonprecise close solution, they would be ineffective if a global solution is sought and are hence inefficient in many practical applications [Rodriguez and Buyya 2014]. Among all heuristic methods, the most powerful of all are the population-based EC algorithms, which effectively exchange search information among team members. As the problem of cloud resource scheduling is seen NP-hard, its intractability is best tackled by an EC algorithm [Roberge et al. 2013; Shen et al. 2014].

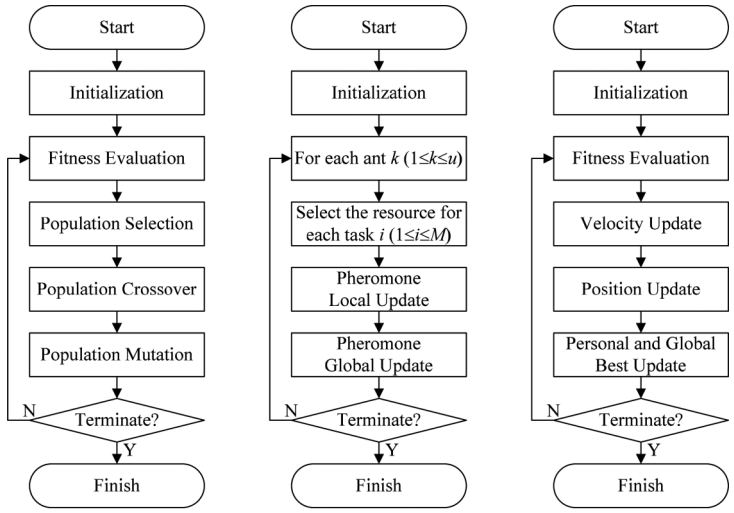


Fig. 5. General algorithmic frameworks of the GA, ACO, and PSO.

EC is a nonconventional optimization paradigm, inspired by the mechanisms of natural evolution and behaviors of living organisms [Zhang et al. 2011c]. In general, EC algorithms include Evolutionary Algorithms (EAs) such as the Genetic Algorithm (GA), swarm intelligence algorithms such as Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), and other nature-inspired algorithms [Yu et al. 2014; Zhan and Zhang 2010; Zhan et al. 2012; Zhang et al. 2014b]. Recent work has shown an emerging trend in the use of EC algorithms for improved effectiveness and efficiency in complex optimization systems [Li et al. 2015d]. This trend would continue with the increase of the proliferation, ambition, and complexity of cloud computing.

Based on the taxonomy of Figure 3, we shall analyze why and how GA, ACO, PSO, and/or other EC algorithms are used for cloud resource scheduling. Various characteristics of different EC methods for various objectives are provided in the next three sections. Algorithm design, scheduling objectives, experimental environments, scales, and results of the work surveyed are also to be listed and compared in tables for the readers to form a quick overview on the EC-based cloud resource scheduling methods. We shall also offer an overview on the advantages and disadvantages of using EC algorithms, following an overview of the EC framework in the next subsection for a wider readership.

2.4. EC Algorithmic Framework and Characteristics

The generic frameworks of the GA, ACO, and PSO in solving the cloud resource scheduling problems are illustrated in Figures 5(a), 5(b), and 5(c), respectively. Their evolutionary processes all go through the cycle of

(i) fitness evaluation,
(ii) candidate selection, and
(iii) trial variation

by a population of candidates searching for potential solutions in parallel and exchanging search information among the candidates. These algorithms are *a posteriori* and nondeterministic in nature. Hence, they require no prior guidance, are suitable for complex spaces, and are able to offer multiple and multiobjective solutions.

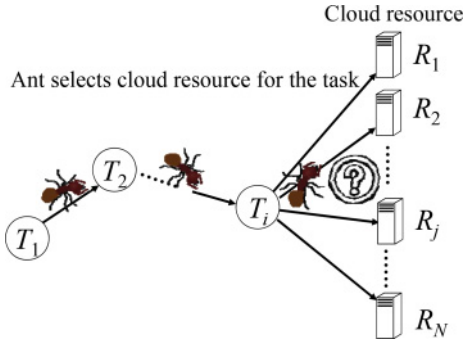


Fig. 6. Illustration of an ant search fashion in ACO for cloud resource scheduling.

Take the flowchart of the GA in Figure 5(a) as an example. A GA regards each cloud scheduling task T_i in Figure 4 as a "gene." The value of the gene is an integer to indicate the resource index of R_j. That is, the task T_i is scheduled to execute on resource R_j. This forms a simple encoding scheme for a "chromosome" that represents a complete set of genes, hence a potential solution. To begin, the GA randomly initializes a population of potential chromosomes. The first step of the optimization process is to evaluate the fitness values of these candidate solutions. Herein the evaluation is based on the optimization objective of the algorithm users. After the evaluation, the GA moves to the second step: selecting better ones to survive to the next generation with their genetic materials (i.e., components of a potential solution) passed on. In the third step, the survived chromosomes will "mate" through "crossover" and then undergo "mutation," which forms new candidates for the next trials. Herein, the reproduction operators can be single-point crossover and random mutation. However, the feasibility of the new solution should be considered. The new generated candidates inherit good information (i.e., genetic materials) from their "parents" a posteriori. This evolutionary cycle iterates until the GA meets a termination condition, which is usually a prespecified number of generations, and then completes the evolution.

For ACO, the algorithm first initializes the performance indicated by "pheromone" in Figure 5(b) and then goes on evolving generation by generation. In each generation, to construct a solution for each "ant," a common strategy is to regard each cloud task T_i as an "ant step," as shown in Figure 6. Then, performance pheromone and heuristic information are adopted to help select the most suitable cloud resource R_j to execute the task. After all the solutions are constructed, pheromone local update and global update are performed. ACO will terminate when a termination condition is met.

For PSO illustrated in Figure 5(c), a general encoding strategy is similar to that in a GA, where each dimension of a "particle" represents a task and the value of the particle indicates the cloud resource index. However, PSO is primarily an optimizer for real values, that is, for the continuous domain. Therefore, the encoding scheme needs to extend to the discrete domain. In each generation, PSO uses the velocity update and position update to guide potential solutions to "fly" toward the globally optimal region. PSO will usually terminate when no significant improvement is found or a maximum number of trials are reached.

Among the three EC algorithms, PSO is often the simplest to implement, with just two explicit equations of position and velocity to be coded. An advantage of EC algorithms is that their computational complexity is not exponentially but (nondeterministic) polynomially related to the problem scale. For a GA, it is practically acceptable that the maximum time taken is proportional to the population size u and the maximum

number G of the generations allowed to run. When taking a fitness evaluation as the elementary operation, all EC algorithms have the same time complexity of $O(uG)$ if there is no exceptional circumstance, as the program overhead is relatively negligible. If we consider the solution construction and update processes, the time complexity varies slightly. For PSO, as each dimension has to be updated, the time complexity would be $O(MuG)$, M being the number of tasks to be scheduled (also the length of the solution). For ACO, as the ant has to select one resource among the N resources for all the M tasks, the time complexity is $O(MNuG)$. Therefore, with the same population size and number of generations allowed, the running time of ACO is longer than that of PSO, which in turn is longer than that of a GA. Nevertheless, they are in the same order, and EC algorithms are more linear for scalability than traditional approaches such as IP, LP, or dynamic programming.

3. SCHEDULING IN THE APPLICATION LAYER

Resource utilization is the ultimate objective of cloud scheduling, as the resources are provided as utility. Therefore, scheduling the cloud resources in the application layer to serve the cloud user is a major topic in cloud resource management and scheduling research. It is twofold when scheduling the cloud resources in this layer. On one hand, it should satisfy the user's QoS and optimize the objectives such as the task makespan (or the complete time), user cost, and application performance. On the other hand, the cloud provider wishes to schedule the cloud resource efficiently to maximize the provision or to save the carbon cost or energy consumed by the cloud center [Buyya et al. 2009; Baliga et al. 2011]. We analyze existing works according to the taxonomy, by dividing them into *scheduling for user QoS*, *scheduling for provider efficiency*, or *scheduling for negotiation* subcategories.

3.1. Scheduling for User QoS

In this subcategory of work, the utilization objective is user QoS, including (i) the makespan, (ii) user costs, (iii) application performance, and (iv) reliability.

Early in 2009, Zhao et al. [2009] proposed the use of a GA scheduler to schedule independent and divisible tasks in a cloud computing environment, with makespan as the objective. As M tasks being scheduled on N cloud resources, a very simple chromosome is shown in Figure 4, with its length the same as the number of the tasks. Each gene $i(1 \leq i \leq M)$ is an integer, for example, $j(1 \leq j \leq N)$, representing the index of the resource, indicating that the ith task T_i is scheduled on the jth resource R_j. Therefore, the GA is used in a very straightforward way and the chromosome coding scheme is typical for scheduling resources.

Such a chromosome coding scheme was also later used by Kumar and Verma [2012], Gan et al. [2010], and Ge and Yuan [2013]. In particular, Kumar and Verm [2012] optimized the makespan by combining a GA with Min-Min and Max-Min strategies to improve population initialization and the GA speed. To enhance the global search capability, Gan et al. [2010] combined simulated annealing with the GA to schedule cloud resources. In their study, not only the makespan was considered, but also were the bandwidth, costs, distances, and reliability. They were combined together with the makespan to form a composite objective function. Ge and Yuan [2013] also scheduled cloud resources with the encoding scheme of Figure 4. They asserted that their GA approach, which considered the total makespan, the average makespan, and the user cost together, was more effective and useful in the cloud computing environment.

Further from independent tasks, tasks on cloud resources can be scheduled with a desired execution order. For example, in the study by Barrett et al. [2011], the tasks have an order and become workflows. Therefore, besides similarity to the coding of Figure 4, the orders of the genes are also considered in the chromosome. Not only

does the value of each gene (indicating which resource hosts the task), but also do the orders of all the genes (indicating the executing order) need to be optimized. The objective function is a combination of the makespan and the user cost. Ye et al. [2011] also scheduled the cloud resources with a GA-based approach by matching the task to the virtual resources and scheduling the execution order of the tasks. In matching the task to the virtual resources, they considered both the VM and the database, as well as the network resources. Therefore, extending from the encoding scheme of Figure 4, each GA gene in the chromosome is coded as a three-dimensional structure with the VM index, database index, and network index. Chen et al. [2015] proposed using dynamic objective strategy to take the makespan as the objective first in order to meet the user deadline constraint, and then to take the cost as the objective to optimize the user benefit.

In addition to optimization for a single or composite objective, multiobjective EC algorithms have also been applied to cloud resource scheduling. For example, Szabo and Kroeger [2012] scheduled resources via a workflow and thus scheduled both VM match and execution orders. They also considered makespan and user costs as two objectives and used the popular NSGA-II multiobjective optimization algorithm to solve this problem.

A general framework of using ACO to schedule user tasks is illustrated in Figure 6, where each ant uses M steps to construct a solution. In the ith step to schedule the ith task T_i, the ant uses pheromone and heuristic information to choose the suitable resource R_j. After M steps, all the M tasks have been scheduled on different resources.

In this context, Banerjee et al. [2009] and Liu et al. [2011] have scheduled the M tasks one by one to the cloud resources, using the scheme in Figure 6, because at each step the task can be scheduled on any resource of a resource set. The pheromone update scheme was modified according to different time slots of cloud service in Banerjee et al. [2009] and the heuristic information based on the user's QoS on user cost, system reliability, response, or security was used to guide the ant to select the optimal resource in Liu et al. [2011]. Zhu et al. [2012] also used an ACO-based approach to schedule cloud resource by considering makespan, user cost, network bandwidth, and system reliability. The tasks were first classified into different categories according to different QoS metrics, and then tasks in different categories were bound to the cloud resources via the ACO optimization.

In addition to GA and ACO algorithms, PSO is also reported in the literature as an effective tool to have scheduled cloud resources successfully. As PSO offers generally faster convergence than other EC algorithms, many researchers have studied scheduling cloud resources by PSO. In the study by Pandey et al. to use PSO for cloud resources scheduling [Pandey et al. 2010], the particle was encoded similarly to the scheme of Figure 4 used in a GA. The particle position length is the same as the task number and the value of each dimension is an integer to indicate the cloud resource that executes the corresponding task. Their approach takes the user cost as the optimization objective, considering both the data transmission cost and computational cost. However, as the velocity update and position update in PSO will make the updated value of each dimension in the position not remain an integer, how to deal with the noninteger issue is not clearly stated in their method.

In order to make PSO suitable for the cloud resource scheduling model, which is a discrete optimization problem, Wu et al. [2010] and Chen and Zhang [2012] extended the continuous PSO to discrete PSO. Their methods are similar in that the set-based mechanism is used. The particle position consists of a set of <task_id, resource_id> pairs. Moreover, the velocity and position updating are modified accordingly to match the code scheme. For the optimization objective, Wu et al. [2010] optimized the user cost by combining the data transmission cost and computational cost, while Chen and

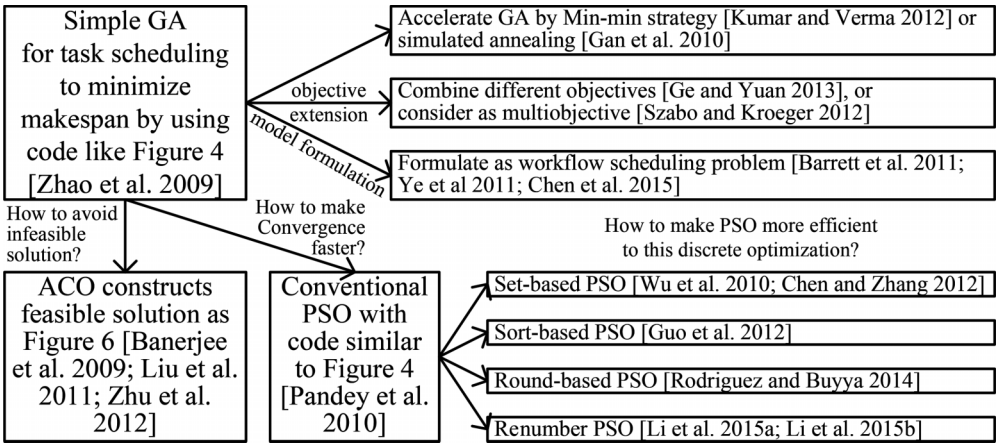


Fig. 7. Development map of using EC to schedule the cloud resources.

Zhang [2012] optimized the user QoS such as makespan, user cost, and reliability separately.

Further extending PSO to the discrete cloud resource scheduling, some other strategies are also proposed. Guo et al. [2012] adopted a small position value rule by sorting all the dimensions in the position according to the real value and giving each dimension an integer-value rank number. This was to transform real-value position to integer-value position, and then to map the integer value to the cloud resource index. In the study, Guo et al. [2012] combined the data transmitting time and user cost to form a single objective. In the study of Rodriguez and Buyya [2014], they rounded the real number to integer number to indicate the resource index that the workflow was scheduled on. However, the index of the resource in fact does not reflect the features of the resources. Therefore, learning from the resources index might make the particles fly randomly. In order to make the guidance more efficient, Li et al. [2015a] proposed a renumber strategy to use the metric of price per unit time to reorder the resources. This way, the learning among particles via resource index becomes much more clear and reasonable. Later, Li et al. [2015b] further extended their work to a multiobjective scheduling model and proposed using coevolutionary multiswarm PSO [Zhan et al. 2013] to solve the problem.

To further compare features and strengths of the previous scheduling approaches, research progress in this field so far is mapped out in Figure 7. It shows how research starts from the simple GA, to "algorithm enhancement," "objective extension," and "model formulation." Relationships of developments from the GA to ACO and to PSO are also mapped out. Moreover, the objectives, general comments on strengths, limitations, and experimental environments, scales, and results of these algorithms are listed and compared in Table II.

3.2. Scheduling for Provider Efficiency

In addition to cloud resource scheduling oriented by the user's QoS, there also exist sizable works on scheduling the cloud resources from the viewpoint of provider efficiency. In this context, the scheduling objectives include (i) load balancing, (ii) utilization maximization, and (iii) minimization of energy consumption. The related works are summarized according to their optimization objectives in Table III and are described as follows.

Table II. Comparisons on Scheduling Approaches for User QoS

Algorithm	Objective	General Comments Strength/Limitation	Experimental Environments	Experimental Scale	Results Compared
GA [Zhao et al. 2009]	Makespan	The algorithmic logic is simple. Only on numerical simulations without a cloud environment	Numerical simulation	2 tasks 2 resources	NA
GA [Kumar and Verma 2012]	Makespan	Inserts two solutions obtained by the Min-Min and Max-Min greedy strategy to accelerate the speed	CloudSim	10–40 tasks 10–40 resources	Traditional GA
GA [Gan et al. 2010]	Makespan, bandwidth, cost, distance, and reliability	Uses SA local search and combines different QoS by weights. No cloud environment or comparisons	Numerical simulation	20 tasks 8 resources	NA
GA [Ge and Yuan 2013]	Makespan, cost	Combines makespan and cost together and uses adaptive strategy for GA parameters	CloudSim	50 tasks 50 resources	CGA and AGA consider only cost/ makespan
GA [Barrett et al. 2011]	Makespan, cost	Considers precedence constraint among tasks; but simple GA may create infeasible solution	CloudSim	20–50 tasks 5–15 resources	Best Resource Selection (BRS)
GA [Ye et al. 2011]	Makespan, cost, availability, and reputation	Considers VM, database, and network resources; combine different QoS by weights to form the objective	Numerical simulation	6–120 tasks 3–20 resources	Exhaustive search and Random selection
GA [Chen et al. 2015]	Makespan, cost	Uses dynamic objective strategy to optimize makespan first and then to optimize the cost	Numerical simulation	100 tasks 30 resources	PSO
GA [Szabo and Kroeger 2012]	Makespan, cost	Models the scheduling as a multiobjective optimization problem and uses the popular NSGA-II	Amazon Cloud	25–1,000 tasks 2–128 resources	GA PSO
ACO [Banerjee et al. 2009]	Makespan	Adds time slot information into pheromone to optimize makespan, but does not consider the fault tolerance issues	Google App Engine, MS Live Mesh	25 kinds of tasks	Traditional ACO Microsoft/ Google resource allocators
ACO [Liu et al. 2011]	Reliability, makespan, cost, and security	Takes different objectives together by weights according to the user QoS, but without comparisons	Numerical simulation	10 tasks 10–50 resources	NA

(Continued)

Table II. Continued

Algorithm	Objective	General Comments Strength/Limitation	Experimental Environments	Experimental Scale	Results Compared
ACO Zhu et al. [2012]	Makespan	Considers priority of the tasks; allows users to define their objectives	CloudSim	20–100 tasks 8 resources	Random distribution algorithm
PSO [Pandey et al. 2010]	Cost	Considers both computation cost and data transmission cost. However, PSO encode does not consider the differences of resources.	Amazon Cloud	5 tasks 3 resources	BRS algorithm
PSO [Wu et al. 2010]	Cost	Uses a randomized adaptive greedy search to initialize feasible solution	Amazon Cloud	50–300 task 3–20 resources	Traditional PSO BRS algorithm
PSO [Chen and Zhang 2012]	Makespan, cost, and reliability	Uses set-based discrete PSO, but only one objective is considered each time	Numerical simulation	9–120 tasks 6–10 resources	Markov decision process ACO
PSO [Rodriguez and Buyya 2014]	Cost	Defines the makespan as deadline constraint. Consider both the computational cost and communication cost.	CloudSim	50–1,000 tasks 6 resources	SCS IC-PCP
PSO [Li et al. 2015a]	Cost	Uses renumber strategy to make the learning among particles efficient	Numerical simulation	200 tasks 10 resources	PSO
PSO [Li et al. 2015b]	Cost	Uses renumber strategy; extend to multiobjective scheduling model	Numerical simulation	100 tasks 10 resources	Renumber PSO

For optimizing the resource load balance, Zhu et al. [2011] proposed using a Multiagent GA (MAGA) to solve the scheduling problem. In order to reduce the size of tasks, a group strategy based on the task parameters was designed to divide the tasks into groups. The MAGA also uses an encoding scheme similar to that of Figure 5, with the length being the same as the number of groups. However, MAGA does not use an integer for each dimension to indicate which VM the corresponding group is scheduled on, but uses a binary code. This is finer for crossover and mutation, but it needs to be converted to an integer when calculating the fitness.

Nishant et al. [2012] also scheduled the cloud resource for an optimized load balance of different nodes, using an ACO-based algorithm. They argued that the ACO approach can first choose the node that has the greatest number of neighboring nodes. This way, the ant can travel in the most possible direction to find more nodes that are overloaded or underloaded. En route, the ant will detect the heavy-load nodes, and redistribute some load to the light-load nodes.

In addition to load balancing, another objective in the provider efficiency oriented scheduling is resource utilization. Lv et al. [2012] proposed using ACO to select proper physical resources for running the users' tasks. Their selection strategy aims to allocate heavy-load resources to the task, so as to save resources. That is, in every ant step their

Table III. Comparisons on Scheduling Approaches for Provider Efficiency

Objective	Algorithm	General Comments Strength/Limitation	Experimental Environments	Experimental Scale	Results Compared
Load balancing	GA [Zhu et al. 2011]	Considers both CPU and memory balance; but use binary encode	Numerical simulation	100 tasks 20 resources	Min_min algorithm
	ACO [Nishant et al. 2012]	A distributed algorithm that ant detects heavy-load node and puts some load to light-load node	No experiments available (NA)	4 resources	NA
Utilization maximization	ACO [Lv et al. 2012]	Ant selects heavy-load resources; but without experiment/ comparison	NA	30 resources	NA
	ACO [Wen et al. 2012]	ACO selects resources, PSO avoids prematurity	Numerical simulation	50–400 tasks	Traditional ACO
Energy consumption minimization	GA [Shen and Zhang 2011]	Uses shadow price strategy to guide evolution	Numerical simulation	500–5,000 tasks 10–50 resources	Traditional GA
	GA [Wang et al. 2012]	GA enhanced with local search	Hadoop	450 tasks 200 resources	Hadoop MapReduce
	ACO [Babukarthik et al. 2012]	ACO as framework and Cuckoo Search (CS) as heuristics; but ignore the task's feature	Xen	1–256 tasks 1–10 resources	Traditional ACO
Multiobjectives	GA [Kessaci et al. 2011]	Energy consumption, CO_2 emission, and profit	Feitelson's PWA	NA	Greedy scheduling

approach does not try to select the light-load resource to make load balance, but to select the heavy-load resource to maximize the resource utilization.

Wen et al. [2012] also improved the cloud resources utilization ratio by scheduling the cloud resources based on a hybrid algorithm consisting of ACO and PSO. Their approach uses ACO as the main process to select suitable resources for different tasks. The pheromone is associated with the resource node. When a resource node is selected for a new arrived task, the pheromone on this node is reduced. They argue that ACO may be trapped into local optimum, and hence hybridize the PSO process to avoid ACO prematurity. In the PSO, however, they did not use real numbers to calculate particle velocities and positions, but used crossover and mutation operations like in a GA to combine the search information of the globally best solution, the personal best solution, and the particle itself.

Another objective for provider efficiency oriented scheduling is to save energy consumption of the cloud center. Shen and Zhang [2011] argued that a conventional GA was slow, especially in large-scale cloud resource scheduling. Therefore, they proposed using a shadow price strategy to guide the GA evolution direction. The shadow price strategy is used to indicate the energy consumption of cloud resources. Their experiments show that their proposed GA-based resource scheduling approach is faster than

the traditional GA-based scheduling approach to reduce energy consumption. Wang et al. [2012] divided the tasks into multijobs that contained "Map tasks" and "Reduce tasks" based on the provider Google's massive data processing framework. These tasks were scheduled by a modified GA with local search to optimize the energy consumption. Babukarthik et al. [2012] proposed a hybrid algorithm that combined the advantages of ACO and Cuckoo Search (CS) to schedule the tasks for saving energy. Their approach uses ACO as the main framework and applies CS instead of heuristic information to find the next resource for the task. However, the motivation and implementation details of the hybrid algorithm are not sufficiently clear in the article.

In the work discussed previously, the researchers have focused on a single optimization objective oriented toward the cloud provider. In the literature, multiobjective optimization for cloud providers was also reported. In the study by Kessaci et al. [2011], a multiobjective GA was used to obtain Pareto solutions to minimal energy consumption, minimal CO2 emission, and maximal provider profits. Their method also used the encoding scheme similar to the one in Figure 4. Therefore, the crossover and mutation operations are relatively straightforward to implement. Moreover, the approach has been evaluated with realistic workload traces from Feitelson's Parallel Workload Archive.

3.3. Scheduling for Negotiation

Users go for cloud computing to enjoy cost savings and avoid resource inefficiencies. However, cloud service providers have varying pricing for computing instances based on different times of the day or SLAs. For instance, an on-demand instance would be more expensive than a reserved instance [Chaisiri et al. 2012]. The user would hence predict their workload or demand patterns in advance, so as not to pay more, and submit their respective workload requests to a "cloud broker" that can aggregate these demands and subsequently distribute the workloads for cost optimization. The research challenge is thus an optimization problem that involves multiple objectives with uncertainly in both ends of the supply and demand chain, where parameters such as pricing, performance, load type, and capacity continuously changes. Although work on scheduling resources in the application layer oriented by user QoS objectives or provider efficiency objectives has gained much attention in recent years, there exist reports focusing on how to efficiently schedule the cloud resource so that both the cloud user's and the cloud provider's objectives can be met in tandem. Here, we term this kind of scheduling as *scheduling oriented by negotiation*. The related works are summarized according to the methods for negotiation in Table IV and are described as follows.

For example, in the study by Jang et al. [2012], a GA-based approach with the same encoding scheme as Figure 4 was used. However, they defined the fitness function by combining the satisfaction of user QoS and the VM availability. Therefore, the scheduling approach can obtain a solution that had good negotiation between cloud users and providers. In addition, Dasgupta et al. [2013] combined the load balance for cloud provider objective and the makespan for cloud user objective. Zhan et al. [2014] enhanced the GA by not only using the Min-Min and Max-Min strategies to generate promising solutions to initialization population, but also by defining the objective that combines both the task makespan and resource load balance.

ACO has also been used in negotiation-oriented scheduling. Chimakurthi and Kumar [2011] argued that the throughput and response time were a concern of cloud users, while the power energy consumption and operational cost were a concern of the cloud providers. Therefore, their power-efficient ACO approach was an adaptive mechanism that can dynamically schedule the resource in cloud computing environments to host services on a minimal number of server resources. This way, not only SLA of the cloud users can be satisfied, but also energy savings of the data center can be achieved.

Table IV. Comparisons on Scheduling Approaches for Negotiation

Method	Algorithm	General Comments Strength/Limitation		Experimental Environments	Experimental Scale	Results Compared
Fitness construction	GA [Jang et al. 2012]	The fitness function combines the objectives of the cloud user and cloud provider	Combine user QoS and VM availability, but do not provide the details	Simulation tool	2,000 tasks 12 resources	Round Robin (RR)
	GA [Dasgupta et al. 2013]		Combine task makespan and resource load balance	CloudAnalyst	25–75 resources	FCFS RR
	GA [Zhan et al. 2014]		Use greedy strategies to produce promising solution in initialization	Numerical simulation	10–40 tasks 10–40 resources	Traditional GA
	ACO [Chimakurthi and Kumar 2011]		A distributed algorithm, but do not provide experimental results	NA	NA	NA
Search guidance	ACO [Li et al. 2011]	The ant chooses the VM for each task step by step by considering both the VM computational ability and load		CloudSim	100–500 tasks 50 resources	FCFS ACO
Two populations	PSO [Copil et al. 2012]	Two populations with objectives of user and provider and exchange solutions sometimes, but without comparisons		Numerical simulation	NA	NA
Equilibrium theory	GA [You et al. 2009]	Approximates the equilibrium state to meet the negotiation of users and providers	The experimental settings are unclear and without comparison	Xen	4 tasks	NA
	Swarm [Yeh et al. 2012]		The experimental scale of tasks and resources are unclear	Numerical simulation	NA	GA PSO
Constraint strategy	GA/ACO/PSO [Wu et al. 2013]	Minimize overall cost while satisfying all constraints from the users and providers		SwinDeW-C	300 tasks 20 resources	GA/ACO/ PSO
Multiobjective	GA [Tao et al. 2014]	Define makespan and energy consumption as the multiobjectives of users and providers		Numerical simulation	5–30 tasks 4 resources	Other GAs

When making a negotiation in ACO, another strategy is to guide the search by considering the negotiation between the cloud user and the provider. Li et al. [2011] proposed scheduling the cloud resource oriented by the objectives of minimizing the makespan of the tasks set (for the cloud users) and optimizing the load balance of the entire system (for the cloud provider). Therefore, in their ACO algorithm, the ant chooses the VM for each task step by step similar to the process shown in Figure 6, but the pheromone for selecting VM is the combination of the VM computing capacity and its current load.

Another example of negotiation-oriented scheduling between the cloud user and the cloud provider was reported in the study by Copil et al. [2012], where the energy consumed by the cloud provider and the performance obtained by the cloud user were regarded as twofold objectives when scheduling the cloud resource within an SLA negotiation. Therefore, it was proposed using a two-population-based PSO approach to the objectives oriented by the cloud user and by the cloud provider separately. These two populations exchange solutions with each other generation by generation, hence facilitating negotiations between the cloud user and the provider.

Some researchers scheduled the resources by introducing an equilibrium theory to the negotiation. In the study by You et al. [2009], an equilibrium state based on a market mechanism balancing the cloud user demand and cloud provider supply was defined. They proposed the use of a GA-based price adjusting algorithm to solve the cloud resource scheduling problem of approximating the equilibrium state to improve resource utilization and to maximize the benefits of both cloud providers and users. Yeh et al. [2012] also defined a Stackelberg equilibrium from the economic community to formulate the bilevel programming that can be used to solve decentralized decision-making problems arising from cloud providers and cloud users. They proposed a simplified swarm optimization algorithm to find the Stackelberg equilibrium solution.

Recently, Wu et al. [2013] proposed using constraint strategy, while Tao et al. [2014] proposed using multiobjectives to satisfy the negotiation between cloud users and providers when scheduling the resources. Wu et al. [2013] argued that in a market-oriented cloud computing environment, the satisfaction of QoS from users, the optimization of profit for the providers, and other factors all should be taken into consideration when scheduling the resources. They designed GA-, ACO-, and PSO-based metaheuristic approaches to scheduling the cloud resources so as to minimize the overall running cost while satisfying all the constraints from the cloud users and providers. Tao et al. [2014] defined makespan and energy consumption as the multiobjectives of cloud user and provider, and designed a GA with a multiparent crossover operator to search for promising Pareto solutions.

4. SCHEDULING IN THE VIRTUALIZATION LAYER

Scheduling for cloud resource utilization is to allocate resources for user tasks so as to optimize the objective of the cloud user, the cloud provider, or both, which is also often coupled with scheduling for resource virtualization. To facilitate the cloud resources as utility provided over the Internet, the physical resources (e.g., the computational resource, storage resource, and network resource) are better virtualized as uniform resources, that is, the VM resources [Xu et al. 2014]. Therefore, how to allocate and migrate VMs on the physical resources becomes a significant research topic [Papagianni et al. 2013; Xiao et al. 2013; Zaman and Grosu 2013]. In the literature, some works argue that VM resource scheduling should aim at the load balance of physical resources, while some argue that allocating as many VMs as possible to a minimal number of physical nodes to save energy should be the major aim. There also exist works on taking the resource virtualization cost as the main consideration. Here, we compare in Table V and classify these works on scheduling for cloud resource virtualization

Table V. Comparisons on Methods for Scheduling at the Virtualization Layer

Category	Algorithm	Objective	General Comments Strength/Limitation	Online/ Offline	Experimental Environments	Experimental Scale	Results Compared
Scheduling for load balance	GA [Zhao et al. 2011]	CPU/memory/ bandwidth balance	Use multiobjective NSGA-II algorithm	Offline	Numerical simulation	10 VMs 6 PMs	Random/Static/ Rank
	GA [Zhou et al. 2012]	Storage resource balance	The solution encode scheme is not clear	Online	Orthrus	4 VMs 2 volume	Orthrus
	GA [Hu et al. 2010]	PM balance, less VM migration	GA with tree code	Online	OpenNebula	15 VMs 5 PMs	Least-load/ Rotating schedule
	ACO [Lu and Gu 2011]	Physical resource balance	ACO finds a load-free node, no comparison	Online	Xen	4 PMs	NA
Scheduling for energy conservation	GA [Zhong et al. 2010]	Reduce the number of PMs	GA with integer code like Figure 4	Offline/ Online	Numerical simulation	5 VMs	First-Fit/RR/ Traditional GA
	GA [Chen et al. 2012]		There are no results comparisons	Online	Numerical simulation	200 VMs 100 PMs	NA
	GA [He et al. 2011]		Offline in initialization and online reconstruct	Offline/ Online	Numerical simulation	50–200 PMs	Entropy algorithm
	GA [Tang and Pan 2014]	Reduce PM energy and communication	The algorithm is scalable	Offline	Numerical simulation	40–120 VMs 17–41 PMs	Traditional GA
	GA [Nakada et al. 2009]	Reduce PMs, minimize migration times	GA, each gene is the number of VMs on this physical node	Online	Xen	NA	NA
	GA [Mi et al. 2010]	Increase resource utilization, less energy consumption	GA with a predictor	Online	Private cloud	300 PMs	TSSP07
	GA [Apostol et al. 2011]	Increase resource utilization	GA: <VM_i, PM_j, t>	Offline	OpenNebula	2–20 VMs 10 PMs	Haizea
	ACO [Feller et al. 2011; Feller and Morin 2012]	Reduce number of PMs	ACO schedules a set of VMs on a set of PMs	Offline	Java simulation	100–600 VMs	First-Fit Decreasing (FFD) CPLEX
	ACO [Gao et al. 2013]	Reduce resource waste and energy consumption	Use multiobjective algorithm, only with numerical simulation	Offline	Numerical simulation	100–2,000 VMs	GA FFD ACO
	ACO [Liu et al. 2014]	Reduce number of PM	Deposit pheromone on VMs pair	Offline	Numerical simulation	100–600 VMs	FFD
Scheduling for cost effectiveness	Mark et al. [2011]	Minimize the cost	Hybrid GA/ACO/PSO	Offline	Numerical simulation	3 VMs 4 PMs	Stochastic IP algorithm
	GA [Lee et al. 2011]	Reduce the task makespan	GA with a prediction engine	Offline	HP Labs Open Cirrus cluster	80 VMs	RR-R H-2
	GA [Rao and Cornelio 2012]	Reduce the task makespan	GA with evolution strategy	Offline	Numerical simulation	NA	NA

to *load balance oriented scheduling*, *energy awareness oriented scheduling*, and *cost saving oriented scheduling*, respectively.

4.1. Scheduling for Load Balance

The scheduling problem of mapping VM resources to physical resources can be formulated similarly to the one of scheduling user tasks on cloud resources. Hence, when using a GA for scheduling, the chromosome code will be similar to the one shown in Figure 4. The length of the chromosome is the number of the VMs and each gene is an integer indicating which PM the VM is scheduled on. According to this encoding scheme, Zhao et al. [2011] proposed the use of a multiobjective GA approach to scheduling the VM resources on physical resources. They considered the load balance of CPU, memory, and bandwidth together as a multiobjective optimization problem, which was solved using the NSGA-II algorithm.

In addition to the computational resource virtualization, storage resource virtualization also plays a significant role in a cloud computing environment. For this, Zhou et al. [2012] proposed a load balance strategy based on a GA to schedule the virtual storage resources on a block storage system. Their experiments show that their dynamic load balance strategy for virtual storage scheduling can help a cloud storage system, that is, the Orthrus, provide higher I/O performance.

Hu et al. [2010] also used a GA approach to map VMs to physical resources with an optimized load balance, but via a different chromosome code scheme. As each PM can host several VMs, their GA approach uses a tree-type encoding scheme to indicate how many VMs are created in each PM. During the cloud runtime, the number of VMs may change (e.g., a new VM may be created or an existing VM may become unavailable). The computational requirements may also change. When these occur, the GA approach remaps the relationship between VMs and PM for a maximized load balance and a minimized VM migration. The authors evaluated and compared their approach in the OpenNebula cloud platform.

In addition to GA-based methods, ACO-based approaches have also been reported for scheduling VM resources on physical resources for cloud resource virtualization. For example, Lu and Gu [2011] proposed using an ACO to find the nearest load-free cloud resource rapidly and to share some load of the overload cloud resources adaptively. In their study, the approach does not schedule a set of VMs on different cloud resources, but just performs the ACO process when some VMs are overloaded. The ACO process just acts like the ant to search for food in order to find the optimal physical resource (e.g., the load-free node) and to create new VMs to share the load of the overloaded VMs.

Analyzing these publications we can conclude that if the user is concerned only with PM resource balancing, an ACO approach is preferred [Lu and Gu 2011]. If the user is concerned only with storage balancing, a GA approach is preferred [Zhou et al. 2012]. If VM migration and resource balancing are both considered, the GA approach proposed by Hu et al. [2010] is preferable. A multiobjective GA is preferred if several resource balancing objectives are considered simultaneously [Zhao et al. 2011].

4.2. Scheduling for Energy Conservation

Different from load balance oriented scheduling, work on energy-aware scheduling aims to schedule VM resources for the minimal number of physical resources. This way, the energy consumption in the cloud center can be reduced. Therefore, we classify works on energy or carbon awareness oriented scheduling into works on maximizing the resource utilization and works on directly reducing the consumed energy or the carbon footprint.

For maximizing the resource utilization, Zhong et al. [2010] designed a super GA scheduler that was able to allocate the VMs efficiently, permitting the maximum usage of physical resource. Therefore, the used PMs were on the maximal utilization ratio and the number of PMs was reduced. The encoding scheme of their approach is similar to that of Figure 4, with the length of the chromosome being the number of VMs and each gene being an integer indicating which PM a VM is scheduled on. They argued that this scheduling problem could be regarded as an unbalance assignment problem, where the optimization objective was to maximize the resource utilization of the used PMs to reduce the number of physical resources and little attention was paid to the load balance of the resources. The GA scheduler operates once in the beginning of the cloud computing system, and will be started every time when the number of VMs or the physical resources changes.

Chen et al. [2012] also used the encoding scheme of Figure 4 to map the VMs to the PMs. Although each gene represents a VM, the genes are ordered according to the memory size of their corresponding VMs. This way, if a chromosome is invalid, for example, when the memory size of the VMs scheduled on the same PM exceeds the physical memory size, the approach can use a greedy strategy to move some VMs to other PMs. Their approach was reported to have effectively achieved the goal of raising physical resource utilization and therefore reducing the number of PMs to save energy. He et al. [2011] also used the VM-to-node map to schedule all VMs to the minimal number of PMs, so as to reduce resource consumption. Therefore, their GA approach was not only used to find an optimized state to map the VMs efficiently to the PMs, but also was extended to reconstruct the system state at a low transition overhead when the environment did change. Tang and Pan [2014] used a hybrid GA with a local search procedure to reduce the energy consumption by considering not only the PMs, but also the communications among VMs.

There also exist other related chromosome encoding schemes in the literature. For example, in the study by Nakada et al. [2009], the chromosome length is set the same as the number of PMs and each gene stores the number (being zero, one, or many) of VMs. They used a VM packing technique to minimize the number of running PMs by using a GA. Mi et al. [2010] used a binary encoding scheme to represent a GA chromosome, which consists of several binary strings with each denoting one kind of VM. Each binary string contains several substrings (being the same number as the PMs), with each substring standing for the average request number that the PM can respond to the corresponding VM. The optimization objective is to find a good allocation solution to host the VM according to the application requirements by minimizing the number of PMs to save energy. Viewing that a GA may be slow to find a solution for reconfiguring the VMs, they proposed to predict future workloads with Brown's quadratic exponential smoothing to help GA find a solution more rapidly. On the other hand, Apostol et al. [2011] considered the scenario that the VM resources and/or the physical resources could change in a different time. Therefore, they designed an encoding scheme for the GA chromosome, where the ith gene was a triplet $<$VM_i, PM_j, $t>$, indicating that the ith VM was scheduled on the jth PM when the time value t was met. The optimization objective in their study is to maximize the resource utilization.

Feller et al. [2011] scheduled the VMs on PMs via the ACO approach to reduce the number of the physical resources. In their study, the resource scheduling problem was modeled as a Multidimensional Bin-Packing (MDBP) problem. In every step of the ACO approach, an ant attempts to choose the next VM (similar to the item in MDBP) efficiently to place on the current physical node (being similar to the bin in MDBP) according to the heuristic and pheromone information. The objective is to place as many VMs on every physical node as possible, so as to reduce the number of active physical nodes to save energy. The authors further extended their study by using an

ACO approach for VMs autonomous and scheduling to save energy in large-scale cloud infrastructures [Feller and Morin 2012]. Similar work to schedule the VMs on PMs via the ACO approach was also conducted by Gao et al. [2013]. However, they defined the total resource wastage and energy consumption as two objectives and proposed solving the problem by multiobjective ACO algorithm. Liu et al. [2014] also proposed an ACO-based approach to reduce the number of PMs. The novelty is that the pheromone is deposited on the VM pair to measure the efficiency of replacing these two VMs together on the same PM.

4.3. Scheduling for Cost Effectiveness

There also exist works showing that the availabilities and prices of different physical resources would be different. Therefore, it is worthwhile optimally to schedule VM resources on physical resources to reduce the total cost. For example, Mark et al. [2011] proposed an evolutionary optimal VM placement algorithm with a demand forecaster to predict the computational demands of the cloud users. They used a hybridized algorithm consisting of a GA, ACO, and PSO to schedule VMs to physical resources efficiently. Their goal was to guarantee the lowest cost, given the user demand, provider's resource availability, and resource price, with optimized VM placement based on the user's usage history.

In addition to direct cost savings, some researchers argue that minimizing the application makespan can lead to indirect cost savings. For example, Lee et al. [2011] adopted a GA approach to scheduling the VM resources using topological information, as the number of PMs was much larger than that of VMs. Therefore, the chromosome was a binary string with the length being the same as the number of the PMs and the binary value of each bit represented whether the PM was selected to host a VM or not. Moreover, a prediction engine was used to utilize the topological information and estimate the performance of a candidate scheduling solution. Their objective was to reduce the application makespan, which would indirectly save the cost. Rao and Cornelio [2012] also carried out work for a similar scheduling problem, and enhanced the GA with an evolution strategy approach. However, the experimental results and comparisons are not detailed.

5. SCHEDULING IN THE DEPLOYMENT LAYER

In addition to scheduling in the application layer and the virtualization layer, another challenging task in cloud resource scheduling is efficient resource deployment. In order to deliver the infrastructure, platform, and software as a service, that is, IaaS, PaaS, and SaaS, to cloud users, cloud providers need to build clusters of servers or even a number of cloud centers located across the country or the whole world to serve worldwide users. Therefore, service resources need to be efficiently deployed in various locations of cloud centers [Larumbe and Sanso 2013]. Moreover, different cloud providers can be grouped together to form a cloud federation that delivers more efficient cloud services or a more balanced cloud center workload, or save energy. This is a "partner federation oriented scheduling" for cloud resource deployment. Here, we use the proposed taxonomy to classify these works on scheduling in the deployment layer as *scheduling for service placement*, *scheduling for partner federation*, and *scheduling for data routing*. Moreover, these related works are summarized and generally commented on in Table VI.

5.1. Scheduling for Service Placement

For service placement oriented scheduling, Yusoh and Tang [2010a] focused their attention on SaaS placement optimization. Their study argued that as Amazon has its storage server resources in America and Europe, while Nirvanix deploys its storage

Table VI. Comparisons on Methods for Scheduling at the Deployment Layer

Classification	Algorithms and General Comments
Scheduling for service placement (How to deploy SaaS/PaaS/IaaS?)	GA for SaaS placement [Yusoh and Tang 2010a]: optimally place the SaaS software/data components to different data centers Enhance GA by parallelism [Yusoh and Tang 2010b; Tang and Yusoh 2012], simulated annealing [Yuan and Wu 2012], and cooperative coevolutionary strategy [Yusoh and Tang 2012a; Yusoh and Tang 2012b]
	GA for PaaS placement to save energy [Agostinho et al. 2011] ACO to place PaaS for load balance [Csorba et al. 2010]
	GA for IaaS (storage) placement [Jindarak and Uthayopas 2011; Guo and Wang 2013]
Scheduling for partner federation (How to federate cloud providers?)	Multiobjective GA, binary code to indicate whether a cloud provider is selected to the federation [Song et al. 2009]
	ACO distributed schedule resources of various cloud providers in the federation, so as to balance the dynamic load of providers [Zhang and Zhang 2010]
	Multiobjective GA selects a partner to carry a service, balance energy consumption and response time [Phan et al. 2012]
Scheduling for data routing (How to find cloud resources fast?)	ACO is used to find the cloud data and services rapidly and effectively [Liu and Huang 2010]
	Hybrid GA and ACO to design the cloud database route scheduling algorithm [Zhang et al. 2011b]
	Hybrid GA and ABC to design the cloud database route scheduling algorithm [Wu and Chen 2011]

services in America, Germany, and Singapore, a SaaS placement problem arises because the software components and data components of the SaaS should be strategically deployed on the worldwide cloud resources, for example, the computational servers resources and the storage servers resources. In their study, each gene is a triplet $<Con, Reg, Ser>$ to represent a component, where *Con*, *Reg*, *Ser* are the IDs of continent, region, and server, respectively. This is, in fact, similar to the encoding scheme in Figure 4, but further takes the geographical position of the cloud resources into consideration. As the code may cause infeasible solution, for example, if *Reg* is larger than *Con*, a penalty is imposed on the fitness of the infeasible chromosomes in scheduling the SaaS components on different cloud resources. The optimization objective is the estimated total execution time for the SaaS components.

The authors later decomposed the scheduling problem into two interacting subproblems and proposed to solve them by using cooperative coevolutionary GAs [Yusoh and Tang 2010b]. One subproblem is to schedule the placement of SaaS's software components to computational servers, while the other is to schedule the placement of SaaS data components to storage servers. The chromosome encoding scheme was still the same as that used in Yusoh and Tang [2010a]. In order to accelerate the evolutionary speed, Tang and Yusoh [2012] further enhanced the GA approach with a parallel cooperative coevolutionary strategy, while Yuan and Wu [2012] proposed using an adaptive simulated annealing GA to schedule the globally distributed data center resources for the SaaS placement.

Yusoh and Tang [2012a] further reported that the dynamic characteristics of the cloud computing environment required dynamic resource scheduling for the SaaS service placement. They proposed a grouping GA approach to cater to the structural group of a composite SaaS and to reconfigure the placement of the SaaS's components. Moreover, the cooperative coevolutionary GA for the SaaS initial placement problem and a repair-based grouping GA for the SaaS resource optimization problem were further developed by Yusoh and Tang [2012b].

In addition to the service placement scheduling in SaaS, work on service placement scheduling in PaaS and IaaS was also reported in the literature. Agostinho et al.

[2011] proposed using a GA-based bio-inspired approach to scheduling the VM services distribution across federated cloud domains, which can be regarded as PaaS service placement scheduling. The optimization objective is to improve energy savings and load balancing in large cloud data centers. For such PaaS service placement scheduling for deploying the VMs, Csorba et al. [2010] proposed an ACO approach for deploying the VM images to the server clusters that reside in different parts of the network. The objective is to balance the load of private and public clouds. Differently, Jindarak and Uthayopas [2011], and Guo and Wang [2013] proposed the GA-based approach to optimally schedule data placements in the cloud storage, which can be regarded as IaaS service placement scheduling. The optimization objective in Jindarak and Uthayopas [2011] is to improve the average data access time for users and the workload balance of the cloud systems, while the objective in Guo and Wang [2013] is to reduce the distributed cooperation costs.

5.2. Scheduling for Partner Federation

In a recent cloud computing environment, a partner federation that allows groups of cloud service providers to collaborate with each other and to publish their services as a single service to the cloud users has become appealing. Song et al. [2009] modeled a cloud partner federation problem as a multitask and multiobjective optimization problem, and proposed using a multiobjective GA to solve this problem. In a chromosome, a binary code is used to represent whether to select a cloud partner in the federation group or not. The multiobjectives include the total price, reliability of services, and relationship between the providers.

In Zhang and Zhang's work [Zhang and Zhang 2010], an approach based on ACO and complexity theory was developed to schedule the resources provided by various service providers in the open cloud computing federation. The objective is to best solve the complex and dynamic load balancing problem in the cloud federation, and to maximize the ratio of user satisfaction and facility utilization among the clouds.

Besides scheduling cloud resources among the cloud partner federation to improve the service quality and to balance workload, work on saving energy in deployment is also reported. Phan et al. [2012] made decisions of service migration and placement with an evolutionary multiobjective GA. The chromosome code is of the same length as the number of the services, and each gene stands for the index of the cloud provider in the federation. In order to provide green clouds, a Pareto-optimal solution balancing trade-offs among renewable energy consumption, cooling energy consumption, and response time performance are optimized.

5.3. Scheduling for Data Routing

Data routing scheduling can be regarded as cloud resource scheduling to access the cloud data and services efficiently after the cloud resources have been deployed. For this, Liu and Huang [2010] argued that the database in cloud computing was required rapidly and effectively. Therefore, how the clouds quickly find the most suitable reasonable database is significant, but this is a difficult task. As ACO is an efficient approach to find the shortest path (routing), the authors proposed an ACO-based data routing approach to rapidly and effectively find the database in the cloud computing environment.

Zhang et al. [2011b] combined the GA and ACO to design a cloud database route scheduling algorithm. Their experiments showed that the approach was reasonable because it could find the required database rapidly and effectively, reducing the dynamical load of cloud database routing and improving the efficiency of cloud computing. Wu and Chen [2011] also proposed finding resources quickly from cloud databases via a GA and ABC fusion approach.

6. RESEARCH CHALLENGES AND EMERGING DIRECTIONS

The promising performance of EC algorithms in scheduling optimization has attracted increasing attention in cloud resource scheduling recently. While much work has been undertaken and studies have been surveyed in the previous sections, it is found that research in this area is still challenging. Many issues remain unexplored and could develop into emerging research directions. In this section, we shall discuss these issues and highlight several potential directions.

6.1. Real-Time Scheduling

In a cloud computing environment, the cloud resources and the user requests can change dynamically. Therefore, a scheduling approach should be smart enough to make real-time responses to a changing environment.

Most of the existing EC works in scheduling cloud resources are centralized approaches because they consider the scheduling problem as a whole and find an optimal solution offline. Although centralized algorithms can use global information to help the scheduling, they often face difficulties online for a real-time response to environmental changes. So far, real-time optimization is still a challenge to EC algorithms because of the population- and iteration-based characteristics of EC algorithms, despite efforts made on real-time algorithm design [Roberge et al. 2013].

One way to tackle such a challenge is to adapt to an online micro EC algorithm with a tiny population size or to augment with local learning on top of an offline solution. However, as cloud scheduling can face challenges of a high volume, velocity, and variety of data arriving during the runtime, a more efficient use of search information and problem information during the evolutionary process should be explored [Zhang et al. 2011c; Zhan et al. 2011]. Moreover, machine learning, data mining, and certain other techniques can be used on these data to predict the cloud user's requirements, cloud resource variations, and other characteristics to guide the search ahead or faster for more efficient cloud resource scheduling.

6.2. Adaptive Dynamic Scheduling

As there are various cloud users with various QoS for the cloud resource utilization, a good scheduling approach should have adaptability to adjust the scheduling process according to dynamic scheduling requirements in the cloud environment. A micro EC algorithm or local learning can help in this regard.

However, the dynamic requirements may come from different aspects. For example, the scheduling objective may change during the system runtime due to changing or different preferences of the cloud users and/or cloud providers. Moreover, the uncertain characteristics of cloud tasks and the addition or removal of some cloud resources often make the scheduling environment change. As such, elasticity is a key feature of cloud resources [Galante and Bona 2012]. Elastic resource provisioning requires scheduling algorithms to work on dynamic topologies, infrastructures, and in real time. Therefore, adaptive control of the EC algorithm according to the environment is a key research issue in developing EC algorithms for complex scheduling in cloud computing. Although it has been proposed to schedule cloud resources by considering the elasticity of the cloud environment [El Zant et al. 2014; Zhang et al. 2011a; Tsoumakos et al. 2013], research in using EC algorithms is challenging due to the real-time requirement, but the development of targeted micro and parallel algorithms should be a promising topic.

EC approaches formulated this way should be able to adjust the algorithmic parameters and operators adaptively and accommodate changing search requirements of the dynamic environments. Adaptive control of EC algorithms has brought high

performance to resource scheduling [Zhan et al. 2009; Di et al. 2014]. It should be noted that adaptive control of EC algorithms can also help with real-time scheduling.

6.3. Large-Scale Scheduling

With the rapid progress of cloud computing, the scales of resources, users, tasks, and workflows have continuously grown. In the future, more and more tasks will be processed in the cloud environment and more and more cloud resources need to be managed and scheduled in the Internet-pervasive environment. Such large-scale cloud resources and demands challenge the current scheduling approaches. Research on developing algorithms for large-scale cloud computing scheduling will emerge as a new trend. A challenge of large-scale scheduling can be that the search landscape of the optimization problem becomes too complex, for example, with too many local optima, that scheduling approaches cannot efficiently find good global solutions.

Fortunately, EC is a promising paradigm for large-scale scheduling, because it offers acceptable solutions in (nondeterministic) polynomial time, although how to design a time-efficient large-scale EC algorithm is still an open problem in the EC community [Li and Yao 2012]. In future research, designing efficient partition strategies to divide large-scale resources into smaller ones and scheduling these small-scale problems with coevolution [Yusoh and Tang 2010b; Zhang et al. 2011c] could be a promising way forward. Moreover, integrating characteristics of cloud resources into algorithm design could also be a useful means.

6.4. Multiobjective Scheduling

A multiobjective nature is inherent in cloud resource scheduling, as the objectives of cloud providers, cloud users, and other stakeholders can be independent. For example, a cloud provider may try to minimize the resource cost via more efficient scheduling, while cloud users are concerned with QoS. Even among cloud users themselves, different users or the same user at different times may have different QoS requirements, such as low computational costs, high speeds, and so on. Therefore, a multiobjective characteristic of the resource scheduling problem could become more and more significant in the future of cloud computing.

Most of the current EC works in scheduling cloud resources consider only one optimization objective or combine multiple objectives into a composite one with weighting. This is acceptable so far, but may need to be addressed in future industrial applications.

It is fortunate that the EC paradigm has proved to be one of the most significant and promising tools for multiobjective optimization [Zhan et al. 2013; Mukhopadhyay et al. 2014]. We believe that in the near future, modeling cloud resources scheduling as a multiobjective problem will naturally become a trend and the use of EC algorithms to solve such a multiobjective problem will accelerate.

6.5. Distributed and Parallel Scheduling

Another challenge in cloud resource scheduling is that the deployment of cloud resources is often distributed with different data centers or at different locations within a cloud center [Sharkh et al. 2013], which is especially the case in interconnected cloud computing [Toosi 2014]. Therefore, distributed processes are necessary. Moreover, cloud users can be from anywhere in the world through the Internet. An intuition is that it would be inefficient to schedule the resources in, for example, Asian data centers to European users or to schedule the resources in European data centers to American users. Therefore, distributed EC algorithms are in need [Tan et al. 2003], which could be implemented in a similar way to parallel EC algorithms. This is not only useful for scheduling cloud resources among different data centers, but is also useful for scheduling different kinds of resources in the same data center.

Apart from meeting the previous challenges in cloud resource management, the large-scale challenge and real-time demand can also benefit from distributed or parallel scheduling. However, many issues about a distributed process need to be explored further. For example, if we divide the populations into subpopulations and run them in various processors and maintain the algorithm diversity to avoid local optima, coevolution and information share among different subpopulations should be considered. A finer-grain implementation is to distribute all the individuals of the population to different processors and run them in parallel. This would save computational time and hence be useful for real-time scheduling.

6.6. Scheduling Cloud Resources for Big Data and Cyber-Physical Integration

The techniques discussed in the previous five subsections should prepare cloud computing in the era of big data, which is characterized by unprecedented volume, velocity, and variety [Dragland 2013]. It would be interesting and rewarding to investigate the role of EC as a whole in the growth of cloud computing and its resource scheduling in the presence of big data through the Internet. An industrial example is that automation systems via the Internet have been developed fast with potentially massive market demands in cloud computing and big data for "Industry 4.0," where cloud models can offer automation functions as services from a dynamic infrastructure [Givehchi et al. 2014]. Intelligently discovering knowledge, relationships, and trends inherent in big data could help foresee emerging customer needs and wants, and hence help scheduling cloud resources for cyber-physical and smart design-manufacture integrations more objectively and more profitably.

With research for more optimal and powerful cloud computing machinery gaining momentum, there emerges a trend that cloud computing creates big data, while big data demand cloud computing in the quest for a sustainable e-infrastructure [Zhang et al. 2014a; Lin et al. 2013; Choi et al. 2013]. More importantly, "data-driven discovery" or synthesis of science, dubbed the "e-Science," is gradually recognized as the "fourth paradigm of science" [Tolle et al. 2011], where in history the experimental science based empirical discovery and description of natural phenomena is regarded as the first paradigm, the theoretical science (e.g., Maxwell's equations) the second, and the computer simulation based "virtual" systems or phenomena the third [Kuhn 2012]. Ultimately, nature-inspired EC will assist cloud computing to play a significant role in finding apparently unconnected pieces of data a relationship between them, leading to data-driven scientific discovery and smart manufacturing, although optimal realization and resource scheduling could require the use of a supercomputer at the present time.

To achieve agile and smart manufacturing, services, or management, cloud resource scheduling inevitably faces new challenges and uncertainties. Most of the challenges discussed so far have to be taken into consideration. The use of metaheuristics such as EC helps holistically identify what data may be classified as inputs and outputs from the cloud and identify how inputs and feedback of outputs may in turn influence the cloud scheduling requirements. Deep learning in EC algorithms for predictive data analytics for cloud scheduling would be one significant topic in the Industry 4.0 era [Lee et al. 2014].

7. CONCLUSIONS

Through analyzing the cloud computing architecture for resource scheduling, this article has first presented the taxonomy of managing and scheduling cloud resources. The taxonomy consists of three categories: *scheduling in the application layer*, *scheduling in the virtualization layer*, and *scheduling in the deployment layer*. In each category, we have analyzed current works, whereby viewpoints of scheduling objectives are

formed, including cloud-user-oriented objectives, cloud-provider-oriented objectives, and system-oriented objectives.

Following this, the landscape of the cloud resource scheduling problem and its state-of-the-art solutions have been briefly reviewed. Then, a comprehensive survey on EC approaches has been presented systematically according to the two-leveled taxonomy. The rapid emergence of publications in this area indicates that this is a dynamic topic, as reflected in the publication media that is mainly via conferences so far.

Looking forward, challenges and potential future research directions have been discussed and invited, including real-time scheduling, adaptive dynamic scheduling, large-scale scheduling, multiobjective scheduling, and distributed and parallel scheduling. At the dawn of Industry 4.0, scheduling for big data and cyber-physical cloud computing has also been investigated. Research in this area is only in its infancy, where more new problems will emerge with the rapid development of cloud computing, big data, and Internet of Things.

ACKNOWLEDGMENTS

The authors would like to thank Yong Wee Foo, Hadoop Administrator and Manager of Communications and Networks Group, Nanyang Polytechnic, Singapore, and University of Glasgow Singapore, for valuable input on data center operations. The authors are also grateful to the anonymous reviewers for their valuable suggestions, especially for those that help extend the readership to developers and other practitioners.

REFERENCES

L. Agostinho, G. Feliciano, L. Olivi, E. Cardozo, and E. Guimaraes. 2011. A bio-inspired approach to provisioning of virtual resources in federated clouds. In *Proceedings of the IEEE 9th International Conference on Dependable, Autonomic and Secure Computing*. 598–604.

Y. Ajiro and A. Tanaka. 2007. Improving packing algorithms for server consolidation. In *Proceedings of the International Conference for the Computer Measurement Group*, 399–406.

E. Apostol, I. Baluta, A. Gorgoi, and V. Cristea. 2011. Efficient manager for virtualized resource provisioning in cloud systems. In *Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing*. 511–517.

M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, et al. 2010. A view of cloud computing. *Communications of the ACM* 53, 4 (2010), 50–58.

R. G. Babukarthik, R. Raju, and P. Dhavachelvan. 2012. Energy-aware scheduling using hybrid algorithm for cloud computing. In *Proceedings of the 3rd International Conference on Computing Communication & Networking Technologies*. 1–6.

T. Back, M. Emmerich, and O. M. Shir. 2008. Evolutionary algorithms for real world applications. *IEEE Computational Intelligence Magazine* 3, 1 (2008), 64–67.

J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker. 2011. Green cloud computing: Balancing energy in processing, storage, and transport. *Proceedings of the IEEE* 99, 1 (2011), 149–167.

S. Banerjee, I. Mukherjee, and P. K. Mahanti. 2009. Cloud computing initiative using modified ant colony framework. *World Academy of Science, Engineering and Technology* 56 (2009), 221–224.

A. K. Bardsiri and S. M. Hashemi. 2012. A review of workflow scheduling in cloud computing environment. *International Journal of Computer Science and Management Research* 1, 3 (2012), 348–351.

E. Barrett, E. Howley, and J. Duggan. 2011. A learning architecture for scheduling workflow applications in the cloud. In *Proceedings of the 9th IEEE European Conference on Web Services*. 83–90.

R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandicc. 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems* 25 (2009), 599–616.

S. Chaisiri, B. Lee, and D. Niyato. 2012. Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing* 5, 2 (2012), 164–177.

Y. Chawla and M. Bhonsle. 2012. A study on scheduling methods in cloud computing. *International Journal of Emerging Trends & Technology in Computer Science* 1, 3 (2012), 12–17.

N. Chen, W. N. Chen, Y. J. Gong, Z. H. Zhan, J. Zhang, Y. Li, and Y. S. Tan. 2014. An evolutionary algorithm with double-level archives for multiobjective optimization. *IEEE Transactions on Cybernetics*, DOI: 10.1109/TCYB.2014.2360923

S. Chen, J. Wu, and Z. H. Lu. 2012. A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness. In *Proceedings of the IEEE 12th International Conference on Computer and Information Technology*. 177–184.

W. N. Chen and J. Zhang. 2012. A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. 773–778.

Z. G. Chen, K. J. Du, Z. H. Zhan, and J. Zhang. 2015. Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, in press.

L. Chimakurthi and M. Kumar. 2011. Power efficient resource allocation for clouds using ant colony framework. *Arxiv preprint arXiv:1102.2608*, 1–6.

H. Choi, S. H. Lee, and D. I. Park. 2013. Biologic data analysis platform based on the cloud. *International Journal of Bio-Science and Bio-Technology*, 5, 3 (2013), 199–206.

G. Copil, D. Moldovan, I. Salomie, T. Cioara, I. Anghel, and D. Borza. 2012. Cloud SLA negotiation for energy saving—A particle swarm optimization approach. In *Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing*. 289–296.

M. J. Csorba, H. Meling, and P. E. Heegaard. 2010. Ant system for service deployment in private and public clouds. In *Proceedings of the 2nd Workshop on Bio-Inspired Algorithms for Distributed Systems*. 19–28.

K. Dasgupta, B. Mandal, P. Dutta, J. K. Mondal, and S. Dam. 2013. A genetic algorithm (GA) based load balancing strategy for cloud computing. *Procedia Technology* 10 (2013), 340–347.

S. Di, C. L. Wang, and F. Cappello. 2014. Adaptive algorithm for minimizing cloud task length with prediction errors. *IEEE Transactions on Cloud Computing* 2, 2 (2014), 194–207.

M. D. Dikaiakos, G. Pallis, D. Katsaros, P. Mehra, and A. Vakali. 2009. Cloud computing: Distributed Internet computing for IT and scientific research. *IEEE Internet Computing* 13, 5 (2009), 10–13.

A. Dragland. 2013. Big data, for better or worse: 90% of world's data generated over last two years. *Science Daily* (May 2013).

B. El Zant, I. Amigo, and M. Gagnaire. 2014. Federation and revenue sharing in cloud computing environment. In *Proceedings of the IEEE International Conference on Cloud Engineering*. 446–451.

E. Feller and C. Morin. 2012. Autonomous and energy-aware management of large-scale cloud infrastructures. In *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. 2542–2545.

E. Feller, L. Rilling, and C. Morin. 2011. Energy-aware ant colony based workload placement in clouds. In *Proceedings of the 12th IEEE/ACM International Conference on Grid Computing*. 26–33.

I. Foster, Y. Zhao, I. Raicu, and S. Lu. 2008. Cloud computing and grid computing 360-degree compared. In *Proceedings of the Grid Computing Environments Workshop*, 1–10.

G. Galante and L. C. E. D. Bona. 2012. A survey on cloud computing elasticity. In *Proceedings of the IEEE/ACM 5th International Conference on Utility and Cloud Computing*. 263–270.

G. N. Gan, T. L. Huang, and S. Gao. 2010. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In *Proceedings of the International Conference on Intelligent Computing and Integrated Systems*. 60–63.

Y. Q. Gao, H. B. Guan, Z. W. Qi, Y. Hou, L. Liu. 2013. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences* 79 (2013), 1230–1242.

J. W. Ge and Y. S. Yuan. 2013. Research of cloud computing task scheduling algorithm based on improved genetic algorithm. In *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*. 2134–2137.

T. A. L. Genez, L. F. Bittencourt, and E. R. M. Madeira. 2012. Workflow scheduling for SaaS/PaaS cloud providers considering two SLA levels. In *Proceedings of the IEEE Network Operations and Management Symposium*. 906–912.

O. Givehchi, H. Trsek, and J. Jasperneite. 2014. Cloud computing for industrial automation systems—A comprehensive overview. In *Proceedings of the 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation*. 1–4.

T. Grandison, E. M. Maximilien, S. Thorpe, and A. Alba. 2010. Towards a formal definition of a computing cloud. In *Proceedings of the IEEE 6th World Congress on Services*. 191–192.

L. Z. Guo, S. G. P. Zhao, S. G. Shen, and C. Y. Jiang. 2012. A particle swarm optimization for data placement strategy in cloud computing. *Information Engineering and Applications, Lecture Notes in Electrical Engineering*, 323–330.

W. Guo and X. Wang. 2013. A data placement strategy based on genetic algorithm in cloud computing platform. In *Proceedings of the 10th Web Information System and Application*. 369–372.

L. G. He, D. Q. Zou, Z. Zhang, H. Jin, K. Yang, and S. A. Jarvis. 2011. Optimizing resource consumptions in clouds. In *Proceedings of the 12th IEEE/ACM International Conference on Grid Computing*. 42–49.

L. Heilig and S. Vob. 2014. A scientometric analysis of cloud computing literature. *IEEE Transactions on Cloud Computing* 2, 3 (2014), 266–278.

J. H. Hu, J. H. Gu, G. F. Sun, and T. H. Zhao. 2010. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *Proceedings of the 3rd International Symposium on Parallel Architectures, Algorithms and Programming*. 89–96.

S. H. Jang, T. Y. Kim, J. K. Kim, and J. S. Lee. 2012. The study of genetic algorithm-based task scheduling for cloud computing. *International Journal of Control and Automation* 5, 4 (2012), 157–162.

B. Jennings and R. Stadler. 2014. Resource management in clouds: Survey and research challenges. *Journal of Network Systems Management* 1–53.

K. Jindarak and P. Uthayopas. 2011. Performance improvement of cloud storage using a genetic algorithm based placement. In *Proceedings of the 8th International Joint Conference on Computer Science and Software Engineering*. 54–57.

H. Kaur and M. Singh. 2012. Review of various scheduling techniques in cloud computing. *International Journal of Networking & Parallel Computing* 1, 2 (2012).

Y. Kessaci, N. Melab, and E. G. Talbi. 2011. A Pareto-based GA for scheduling HPC applications on distributed cloud infrastructures. In *Proceeding of the International Conference on High Performance Computing and Simulation*. 456–462.

T. S. Kuhn. 2012. *The Structure of Scientific Revolutions*, University of Chicago Press.

P. Kumar and A. Verma. 2012. Independent task scheduling in cloud computing by improved genetic algorithm. *International Journal of Advanced Research in Computer Science and Software Engineering* 2, 5 (2012), 111–114.

S. Kumar and P. Balasubramanie. 2012. Dynamic scheduling for cloud reliability using transportation problem. *Journal of Computer Science* 8, 10 (2012), 1615–1626.

F. Larumbe and B. Sanso. 2013. A tabu search algorithm for the location of data centers and software components in green cloud computing networks. *IEEE Transactions on Cloud Computing* 1, 1 (2013), 22–35.

G. Lee, N. Tolia, P. Ranganathan, and R. H. Katz. 2011. Topology-aware resource allocation for data-intensive workloads. *ACM SIGCOMM Computer Communication Review* 41 (2011), 120–124.

J. Lee, H.-A. Kao, and S. Yang. 2014. Service innovation and smart analytics for Industry 4.0 and big data environment. *Procedia CIRP* 16 (2014), 3–8.

H. H. Li, Y. W. Fu, Z. H. Zhan, and J. J. Li. 2015a. Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling. In *Proceedings of the IEEE Congress on Evolution Computation*, in press.

H. H. Li, Z. G. Chen, Z. H. Zhan, K. J. Du, and J. Zhang. 2015b. Renumber coevolutionary multiswarm particle swarm optimization for multi-objective workflow scheduling on cloud computing environment. In *Proceedings of the Genetic Evolutionary Computation Conference*.

K. Li, G. C. Xu, G. Y. Zhao, Y. S. Dong, and D. Wang. 2011. Cloud task scheduling based on load balancing ant colony optimization. In *Proceedings of the 6th Annual ChinaGrid Conference.* 3–9.

Q. Li and Y. K. Guo. 2010. Optimization of resource scheduling in cloud computing. In *Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. 315–320.

X. D. Li and X. Yao. 2012. Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions on Evolutionary Computation* 16, 2 (2012), 210–224.

Y. H. Li, Z. H. Zhan, S. J. Lin, J. Zhang, and X. N. Luo. 2015c. Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems. *Information Sciences* 293 (2015), 370–382.

Y. L. Li, Z. H. Zhan, Y. J. Gong, W. N. Chen, J. Zhang, and Y. Li. 2014. Differential evolution with an evolution path: A DEEP evolutionary algorithm. *IEEE Transactions on Cybernetics*, DOI: 10.1109/TCYB.2014.2360752

Y. L. Li, Z. H. Zhan, Y. J. Gong, J. Zhang, Y. Li, and Q. Li. 2015d. Fast micro-differential evolution for topological active net optimization, *IEEE Transactions on Cybernetics*, in press.

Y. C. Lin, C. S. Yu, and Y. J. Lin. 2013. Enabling large-scale biomedical analysis in the cloud. *BioMed Research International*, 2013, Article ID 185679, 1–6.

H. Liu, D. Xu, and H. K. Miao. 2011. Ant colony optimization based service flow scheduling with various QoS requirements in cloud computing. In *Proceedings of the 1st ACIS International Symposium on Software and Network Engineering*. 53–58.

J. Liu and T. L. Huang. 2010. Dynamic route scheduling for optimization of cloud database. In *Proceedings of the International Conference on Intelligent Computing and Integrated Systems*. 680–682.

X. F. Liu, Z. H. Zhan, K. J. Du, and W. N. Chen. 2014. Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 41–47.

X. Lu and Z. L. Gu. 2011. A load-adaptive cloud resource scheduling model based on ant colony algorithm. In *Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems*. 296–300.

Q. C. Lv, X. X. Shi, and L. Z. Zhou. 2012. Based on ant colony algorithm for cloud management platform resources scheduling. In *Proceedings of the World Automation Congress*. 1–4.

C. C. T. Mark, D. Niyato, and C. K. Tham. 2011. Evolutionary optimal virtual machine placement demand forecaster for cloud computing. In *Proceedings of the International Conference on Advanced Information Networking and Applications*. 348–355.

H. B. Mi, H. M. Wang, G. Yin, Y. F. Zhou, D. X. Shi, and L. Yuan. 2010. Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Proceedings of the IEEE International Conference on Services Computing*. 514–521.

H. Morshedlou and M. R. Meybodi. 2014. Decreasing impact of SLA violations: A proactive resource allocation approach for cloud computing environments. *IEEE Transactions on Cloud Computing* 2, 2 (2014), 156–167.

A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. Coello Coello. 2014. A survey of multiobjective evolutionary algorithms for data mining: Part I. *IEEE Transactions on Evolutionary Computation* 18, 1 (2014), 4–19.

H. Nakada, T. Hirofuchi, H. Ogawa, and S. Itoh. 2009. Toward virtual machine packing optimization based on genetic algorithm. *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, Lecture Notes in Computer Science, Volume 5518, 651–65.

K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, N. Nitin, and R. Rastogi. 2012. Load balancing of nodes in cloud using ant colony optimization. In *Proceedings of the 14th International Conference on Computer Modelling and Simulation*. 3–8.

S. Pandey, L. L. Wu, S. M. Guru, and R. Buyya. 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networks and Applications*. 400–407.

C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, and A. Monje. 2013. On the optimal allocation of virtual resources in cloud computing networks. *IEEE Transactions on Computers* 62, 6 (2013), 1060–1071.

D. H. Phan, J. Suzuki, R. Carroll, S. Balasubramaniam, W. Donnelly, and D. Botvich. 2012. Evolutionary multiobjective optimization for green clouds. In *Proceedings of the Genetic and Evolutionary Computation Conference* 19–26.

J. J. Rao and K. V. Cornelio. 2012. An optimised resource allocation approach for data-intensive workloads using topology-aware resource allocation. In *Proceedings of the IEEE International Conference on Cloud Computing in Emerging Markets*. 1–4.

M. A. Rodriguez and R. Buyya. 2014. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing* 2, 2 (2014), 222–235.

V. Roberge, M. Tarbouchi, and G. Labonte. 2013. Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Transactions on Industrial Informatics* 9, 1 (2013), 132–141.

M. A. Sharkh, M. Jammal, A. Shami, and A. Ouda. 2013. Resource allocation in a network-based cloud computing environment: Design challenges. *IEEE Communications Magazine* 51, 11 (2013), 46–52.

G. Shen and Y. Q. Zhang. 2011. A shadow price guided genetic algorithm for energy aware task scheduling on cloud computers. In *Proceedings of the International Conference on Advances in Swarm Intelligence*. Lecture Notes in Computer Science, Volume 6728. Springer, Berlin, 522–529.

M. Shen, Z. H. Zhan, W. N. Chen, Y. J. Gong, J. Zhang, and Y. Li. 2014. Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks. *IEEE Transactions on Industrial Electronics* 61, 12 (2014), 7141–7151.

B. Song, M. M. Hassan, E. N. Huh, C. W. Yoon, and H. W. Lee. 2009. A hybrid algorithm for partner selection in market oriented cloud computing. In *Proceedings of the International Conference on Management and Service Science*. 1–4.

B. Speitkamp and M. Bichler. 2010. A mathematical programming approach server consolidation problems in virtualized data centers. *IEEE Transactions on Services Computing* 3, 4 (2010), 266–278.

C. Szabo and T. Kroeger. 2012. Evolving multi-objective strategies for task allocation of scientific workflows on public clouds. In *Proceedings of the IEEE World Congress on Computation Intelligence*. 1–8.

K. C. Tan, A. Tay, and J. Cai. 2003. Design and implementation of a distributed evolutionary computing software. *IEEE Transactions on Systems, Man, and Cybernetics* 33, 3 (2003), 325–338.

M. Tang and S. Pan. 2014. A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers. *Neural Processing Letters*, DOI: 10.1007/s11063-014-9339-8, 1–11

M. Tang and Z. I. M. Yusoh. 2012. A parallel cooperative co-evolutionary genetic algorithm for the composite SaaS placement problem in cloud computing. *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Volume 7492. Springer-Verlag, Berlin, 225–234.

F. Tao, Y. Feng, L. Zhang, and T. W. Liao. 2014. CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Applied Soft Computing* 19 (2014), 264–279.

K. M. Tolle, D. Tansley, and A. J. G. Hey. 2011. The fourth paradigm: Data-intensive scientific discovery. *Proceedings of the IEEE* 99, 8 (2011), 1334–1337.

A. N. Toosi, R. N. Calheiros, and R. Buyya. 2014. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surv*eys 47, 1 (2014), 1–47.

D. Tsoumakos, I. Konstantinou, C. Boumpouka, S. Sioutas, and N. Koziris. 2013. Automated, elastic resource provisioning for NoSQL clusters using TIRAMOLA. In *Proceedings of the IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. 34–41.

H. N. Van, F. D. Tran, and J. Menaud. 2010. Performance and power management for cloud infrastructures. In *Proceedings of the IEEE 3rd International Conference on Cloud Computing*. 329–336.

X. L. Wang, Y. P. Wang, and H. Zhu. 2012. Energy-efficient multi-job scheduling model for cloud computing and its genetic algorithm. *Mathematical Problems in Engineering*, Article ID 589243, 1–16.

X. T. Wen, M. H. Huang, and J. H. Shi. 2012. Study on resources scheduling based on ACO algorithm and PSO algorithm in cloud computing. In *Proceedings of the 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science*. 219–222.

H. J. Wu and S. H. Chen. 2011. Cloud database resource calculations optimization based on buzzers and genetic algorithm double-population evolution mechanism. In *Proceedings of the Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*. 1188–1192.

Z. J. Wu, X. Liu, Z. W. Ni, D. Yuan, and Y. Yang. 2013. A market-oriented hierarchical scheduling strategy in cloud workflow systems. *Journal of Supercomputing* 63, 1 (2013), 256–293.

Z. J. Wu, Z. W. Ni, L. C. Gu, and X. Liu. 2010. A revised discrete particle swarm optimization for cloud workflow scheduling. In *Proceedings of the International Conference on Computational Intelligence and Security*. 184–188.

Z. Xiao, W. Song, and Q. Chen. 2013. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transactions on Parallel and Distributed Systems* 24, 6 (2013), 1107–1117.

F. Xu, F. M. Liu, H. Jin, and A. V. Vasilakos. 2014. Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proceedings of the IEEE* 102, 1 (2014), 11–31.

J. L. Xu, J. Tang, K. Kwiat, W. Y. Zhang, and G. L. Xue. 2013. Enhancing survivability in virtualized data centers: A service-aware approach. *IEEE Journal on Selected Areas in Communications* 31, 12 (2013), 2610–2619.

Z. Ye, X. F. Zhou, and A. Bouguettaya. 2011. Genetic algorithm based QoS-Aware service compositions in cloud computing. *Database Systems for Advanced Applications*, Lecture Notes in Computer Science, Volume 6588. Springer, Berlin, 321–334.

W. C. Yeh, Y. M. Yeh, and L. M. Lin. 2012. The application of bi-level programming with Stackelberg equilibrium in cloud computing based on simplified swarm optimization. In *Proceedings of the 8th International Conference on Computing Technology and Information Management*. 809–814.

X. D. You, X. H. Xu, J. Wan, D. J. Yu. 2009. RAS-M: Resource allocation strategy based on market mechanism in cloud computing. In *Proceedings of the 4th Annual ChinaGrid Conference*. 256–263.

J. Yu, R. Buyya, and K. Ramamohanarao. 2008. Workflow scheduling algorithms for grid computing. *Metaheuristics for Scheduling in Distributed Computing Environments Studies in Computational Intelligence* 146 (2008), 173–214.

W. J. Yu, M. Shen, W. N. Chen, Z. H. Zhan, Y. J. Gong, Y. Lin, O. Liu, and J. Zhang. 2014. Differential evolution with two-level parameter adaptation. *IEEE Transactions on Cybernetics* 44, 7 (2014), 1080–1099.

B. W. Yuan and S. C. Wu. 2012. An adaptive simulated annealing genetic algorithm for the data placement problem in SAAS. In *Proceedings of the International Conference on Industrial Control and Electronics Engineering*. 1037–1043.

Z. I. M. Yusoh and M. Tang. 2010a. A penalty-based genetic algorithm for the composite SaaS placement problem in the cloud. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 1–8.

Z. I. M. Yusoh and M. Tang. 2010b. A cooperative coevolutionary algorithm for the composite SaaS placement problem in the cloud. *Neural Information Processing: Theory and Algorithms*, Lecture Notes in Computer Science, Volume 6443. Springer, Berlin, 618–625.

Z. I. M. Yusoh and M. Tang. 2012a. Composite SaaS placement and resource optimization in cloud computing using evolutionary algorithms. In *Proceedings of the IEEE 5th International Conference on Cloud Computing*. 590–597.

Z. I. M. Yusoh and M. Tang. 2012b. Clustering composite SaaS components in cloud computing using a grouping genetic algorithm. In *Proceedings of the IEEE World Congress on Computational Intelligence*. 1–8.

S. Zaman and D. Grosu. 2013. A combinatorial auction-based mechanism for dynamic VM provisioning and allocation in clouds. *IEEE Transactions on Cloud Computing* 1, 2 (2013), 129–141.

Z. H. Zhan, J. Li, J. Cao, J. Zhang, H. Chung, and Y. H. Shi. 2013. Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems. *IEEE Transactions on Cybernetics* 43, 2 (2013), 445–463.

Z. H. Zhan, G. Y. Zhang, Y. Lin, Y. J. Gong, and J. Zhang. 2014. Load balance aware genetic algorithm for task scheduling in cloud computing. In *Simulated Evolution and Learning*, Lecture Notes in Computer Science, Volume 8886, 644–655.

Z. H. Zhan and J. Zhang. 2010. Self-adaptive differential evolution based on PSO learning strategy. In *Proceedings of the Genetic and Evolutionary Computation Conf*erence. 39–46.

Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung. 2009. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics* 39, 6 (2009), 1362–138.

Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi. 2011. Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation* 15, 6 (2011), 832–847.

Z. H. Zhan, J. Zhang, Y. H. Shi, and H. L. Liu. 2012. A modified brain storm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 1–8.

F. Zhang, J. Cao, K. Hwang, and C. Wu. 2011a. Ordinal optimized scheduling of scientific workflows in elastic compute clouds. In *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science*. 9–17.

F. Zhang, J. Cao, W. Tan, S. U. Khan, K. Li, and A. Y. Zomaya. 2014a. Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud. *IEEE Transactions on Emerging Topics in Computing* 2, 3 (2014), 338–351.

M. D. Zhang, Z. H. Zhan, J. J. Li, and J. Zhang. 2014b. Tournament selection based artificial bee colony algorithm with elitist strategy. In *Proceedings of the Conference on Technologies and Applications of Artificial Intelligence*, 387–396.

Q. Zhang, L. Cheng, and R. Boutaba. 2010. Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 1, 1 (2010), 7–18.

Y. H. Zhang, L. Feng, and Z. Yang. 2011b. Optimization of cloud database route scheduling based on combination of genetic algorithm and ant colony algorithm. *Procedia Engineering* 15 (2011), 3341–3345.

J. Zhang, Z. H. Zhan, Y. Lin, N. Chen, Y. J. Gong, J. H. Zhong, H. Chung, Y. Li, and Y. H. Shi. 2011c. Evolutionary computation meets machine learning: A survey. *IEEE Computational Intelligence Magazine* 6, 4 (2011), 68–75.

Z. H. Zhang and X. J. Zhang. 2010. A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. In *Proceedings of the 2nd International Conference on Industrial Mechatronics and Automation*. 240–243.

C. H. Zhao, S. S. Zhang, Q. F. Liu, J. Xie, and J. C. Hu. 2009. Independent tasks scheduling based on genetic algorithm in cloud computing. In *Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing*, 1–4.

J. F. Zhao, W. H. Zeng, M. Liu, and G. M. Li. 2011. Multi-objective optimization model of virtual resources scheduling under cloud computing and its solution. In *Proceedings of the International Conference on Cloud and Service Computing*. 185–190.

H. Zhong, K. Tao, and X. J. Zhang. 2010. An approach to optimized resource scheduling algorithm for open-source cloud systems. In *Proceedings of the 5th Annual ChinaGrid Conference*. 124–129.

L. Zhou, Y. C. Wang, J. L. Zhang, J. Wan, and Y. J. Ren. 2012. Optimize block-level cloud storage system with load-balance strategy. In *Proceedings of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. 2162–2167.

L. N. Zhu, Q. S. Li, and L. N. He. 2012. Study on cloud computing resource scheduling strategy based on the ant colony optimization algorithm. *International Journal of Computer Science Issues* 9, 5 (2012), 54–58.

K. Zhu, H. G. Song, L. J. Liu, J. Z. Gao, and G. J. Cheng. 2011. Hybrid genetic algorithm for cloud computing applications. In *Proceedings of the IEEE Asia-Pacific Services Computing Conference*. 182–187.

Received July 2014; revised March 2015; accepted May 2015