Metaheuristic Scheduling for Cloud: A Survey

Chun-Wei Tsai and Joel J. P. C. Rodrigues *Senior Member, IEEE*

***Abstract*—Cloud computing has become an increasingly important research topic given the strong evolution and migration of many network services to such computational environment. The problem that arises is related with efficiency management and utilization of the large amounts of computing resources. This paper begins with a brief retrospect of traditional scheduling, followed by a detailed review of metaheuristic algorithms for solving the scheduling problems by placing them in a unified framework. Armed with these two technologies, this paper surveys the most recent literature about metaheuristic scheduling solutions for cloud. In addition to applications using metaheuristics, some important issues and open questions are presented for the reference of future researches on scheduling for cloud.**

***Index Terms*—Cloud computing, metaheuristics, scheduling.**

I. Introduction

THE goal of a scheduler [1]–[5] is to find ways to appropriately assign tasks to limited resources that optimize one or more objectives. It is generally believed that modern scheduling approaches can be dated back to Johnson's study [6]. Nowadays, scheduling is widely used in different applications, such as manufacturing of printed circuit boards, power system control, and scheduling of multimedia data objects on the World Wide Web (WWW) [3]. Since one of the important applications of modern scheduling is the assignment of tasks from users of the Internet to limited resources on distributed computing systems, from the 1980s until now, these systems have undergone several changes. One of the early changes was the emergence of cluster systems that integrate a number of standalone computers together to work as a single system [7], [8]. To overcome the problem of cluster systems being only able to use local resources, the next change, grid, has been developed to combine all the available heterogeneous resources from geographically distributed institutions [9]. A recent change is the shift to cloud computing systems [10]–[13] that leverage the strengths of cluster and grid.

Given the seemingly unlimited computing resources of the new type of computing systems, unfortunately, there exist no polynomial time-scheduling algorithms to optimize the allocation of these computing resources because most scheduling problems are either NP-hard or NP-complete [14]. Taillard [15] presented a simple example to explain the dilemma we are facing, that is, less than 0.02% of the candidate solutions are between the makespan of the optimum solution and 1.01 times the makespan of the optimum solution. This example tells us that it will be very difficult to find the optimum solution for large problems. As a consequence, researchers have focused on seeking a good algorithm to solve scheduling problems.

Two common methods for scheduling on current computer systems are exhaustive algorithm and deterministic algorithm (DA). In practice, DAs [16] are much better than traditional exhaustive algorithms because DAs are faster for scheduling problems. However, the two main disadvantages of DAs are in that they are not designed for all the data distributions, and most DAs are inappropriate for large-scale scheduling problems. Unlike DAs and the exhaustive algorithm, metaheuristic algorithms (also called approximate algorithms) employ iterative strategies to find solutions in a reasonable time. Numerous research results [15], [17]–[20] were presented to show that metaheuristic scheduling algorithms can provide better scheduling results than traditional scheduling algorithms.

However, their focus is not on cloud computing environment. Although many scheduling methods have been shown successful on different computing environments (i.e., grid computing or clustering computing), and some of the ideas of these scheduling methods perhaps can be directly used on cloud computing scheduling, they are not designed for cloud computing and thus may not be the best scheduling strategy. This paper not only provides a systematic description of scheduling but also gives a bridge to associate traditional scheduling with metaheuristic scheduling to provide a guideline for the researchers focusing on traditional scheduling to shift to metaheuristic scheduling on cloud computing systems. Note that metaheuristic algorithms are used in this paper to differentiate traditional heuristic algorithms from modern heuristic algorithms.

The remainder of this paper is organized as follows. Section II begins with a brief introduction to traditional scheduling. The main purpose of Section III is to use a unified metaheuristic framework to associate metaheuristic scheduling algorithms with others to reduce the effort in learning these algorithms. Section IV gives a brief review of metaheuristic scheduling on cloud, which includes the characteristics of scheduling problems, measurements, and algorithms presented to solve them. Conclusions and future trends are drawn in Section V.

Manuscript received August 1, 2012; revised December 19, 2012; accepted March 3, 2013. Date of publication May 16, 2013; date of current version February 5, 2014. This work was supported in part by the National Science Council of Taiwan, under Grant NSC101-2221-E-041-012, Instituto de Telecomunicações, Next Generation Networks and Applications Group (NetGNA), Portugal, and National Funding from the Fundação para a Ciência e a Tecnologia through the PEst-OE/EEI/LA0008/2011 Project.

C.-W. Tsai is with the Department of Applied Informatics and Multimedia, Chia Nan University of Pharmacy & Science, Tainan 717, Taiwan (e-mail: cwtsai0807@gmail.com).

J. J. P. C. Rodrigues is with the Instituto de Telecomunicações, University of Beira Interior, 6201-001 Covilhã, Portugal (e-mail:joeljr@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2013.2256731

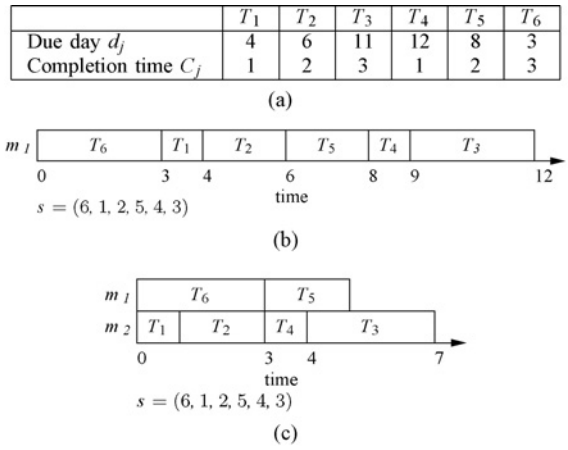

Fig. 1. Simple example illustrating how a set of given tasks are allocated to a set of given machines. (a) Due day and completion time of each task. (b) Results obtained by assigning all the tasks to a single machine. (c) Results obtained by assigning all the tasks to a parallel machine.

II. Traditional Scheduling

The so-called scheduling problem [3], [21], [22] can usually be considered as a problem the objective of which is to allocate a set of given tasks $T = \{T_1, T_2, \ldots, T_n\}$ to a set of given machines $M = \{M_1, M_2, \ldots, M_m\}$ subject to the constraints of optimizing one or more predefined measures or objective functions. When there is one and only one machine, i.e., $m = 1$, the scheduling problem is referred to as a single processor (single machine) scheduling problem. When there is more than one machine, i.e., $m \geq 2$, the scheduling problem is regarded as a multiprocessor (parallel machine) scheduling problem. The objective functions makespan, lateness, tardiness, flowtime, and their variants are widely used in scheduling studies to measure the performance of scheduling algorithms. For the details and other measurement methods, readers are referred to [1] and [22].

A simple example is given in Fig. 1 to make the idea more concrete. The example shows the results of assigning a set of six tasks, with the due day and completion time of each task given in Fig. 1(a), to a single machine and a parallel machine. In the case of a single processor (single machine), the results are as shown in Fig. 1(b). That is, with $n = 6$ and $m = 1$, if the objective function is makespan, then $C_{\max} = 12$ for the given solution (6, 1, 2, 5, 4, 3). If the objective function is the number of tardy jobs,[1] then the same solution gives $\mathcal{U}_{\text{tot}} = 1$. In the case of a parallel machine, the results are as given in Fig. 1(c), which show that if there are two machines (i.e., $m = 2$) and the same objective function is used, then $C_{\max} = 7$ with the solution (6, 1, 2, 5, 4, 3). However, Fig. 2(a) shows that the makespan $C_{\max}$ can be improved because the total completion time is 12, and the makespan possible is 6. As most studies attempt to adjust the combinations (T_4 and T_3) on the critical path (T_6, T_4, and T_3) [23], the example given in Fig. 2(b) shows that the makespan $C_{\max}$ can be reduced from 7 to 6. These examples illustrate that different constraints and object functions can be used to make the problem conform to the situation we are facing.

[1]That is, the number of tasks that does not meet its due date.

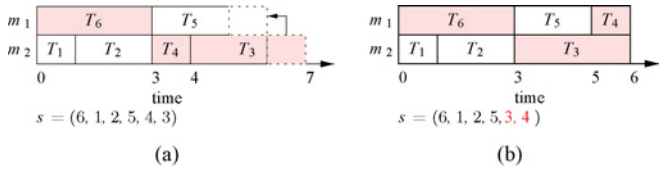

Fig. 2. Simple example used to explain how to improve the scheduling results given in Fig. 1(c).

Because scheduling is widely used in many problem domains, several studies [1], [4], [16], [24], [25] attempted to present a taxonomy of scheduling problems in terms of the description of jobs (e.g., processing time, release date, due date, and weight), machines (e.g., single versus multiple), and other details, such as preemption versus nonpreemption, precedence versus nonprecedence, batch versus nonbatch, sequence-dependent versus sequence-independent, and online versus offline. These are the constraints that are often employed to differentiate and describe scheduling problems. A three-field notation, $\alpha/\beta/\gamma$, was presented in [24] to describe all the scheduling problems. In this notation, α specifies the type of machine, such as single machine, parallel machine, flow shop, and job shop; β specifies the processing characteristics and constraints, such as sequence-independent or sequence-dependent; and γ specifies the measures, such as makespan or the number of tardy (late) jobs. The length of time was used in [16] to classify scheduling into five levels: long-range planning, middle-range planning, short-range planning, scheduling, and reactive scheduling/control. For more details of the traditional scheduling algorithms, readers are referred to [1] in which not only a comprehensive survey of the traditional scheduling algorithms but also valuable comments from various perspectives were given.

III. Metaheuristic Scheduling Algorithms

A. Unified Framework of Metaheuristics

We begin with a framework of metaheuristics, followed by an introduction to the metaheuristic algorithms inspired by evolution program (EP) [26]. Unlike EP, which assumes that the solutions to be passed on to the next iterations are selected by a particular mechanism, the presented framework expands the scope of EP to contain a larger number of metaheuristic algorithms. The number of solutions searched each iteration can be one or more, and the determination operator instead of the selection operator is used to break the limitation by selecting some of the current solutions to pass them on to later iterations.

As Algorithm III-A shows, the initial, transition, evaluation, and determination operators are employed in this framework where s denotes the current solution, v the candidate solution, and f the evaluated value of s. In addition to these operators, how the solution is encoded (represented) is a critical issue for applying metaheuristic algorithms to scheduling problems. Also, to simplify the discussion that follows, **T**, **E**, and **D** will

Algorithm 1 Metaheuristic Algorithm

```
1: Create the initial solution s
2: While the termination criterion is not met
3:     v = Transition(s)            T
4:     f = Evaluation(v)            E
5:     s = Determination(v, f)      D
6: End
```

be used to denote, respectively, the transition, evaluation, and determination mechanisms of each algorithm described in this paper.

1) *Transition:* This operator plays the role of changing[2] the current solution(s) to the next state. Perturbative and constructive [27] are the two common transition methods for combinatorial problems. Depending on the design of metaheuristics, of course, this operator can be very simple or very complex.

2) *Evaluation:* This operator is responsible for evaluating the value of the objection function of the problem in question, such as makespan of a scheduling problem. Some metaheuristics do not employ an objection function to measure the solutions directly. Instead, they use other measurement mechanisms to determine the value in the decision space. Since the value can be evaluated either in the objective space or in the decision space, this implies that the value may not represent the true quality of the solution of the optimization problem.

3) *Determination:* This operator plays the role of guiding the search. That is, this operator determines not only the directions but also the intensification or diversification of the search, which in turn may influence the convergence speed.

The example given in Fig. 3 illustrates how a metaheuristic algorithm for the scheduling problem works, i.e., how the metaheuristic algorithm uses the transition, evaluation, and determination operators to search for the solutions. In this example, the transition operator is a swap operator that is aimed at exchanging subsolutions of the same solution or of different solutions, such as exchanging subsolutions 3 and 4 of solution s_1 and exchanging subsolution 4 of solution s_1 and subsolution 3 of solution s_2. Exchanging subsolutions of solutions usually needs a repair mechanism to make sure that the solution is legal for the problem after they are exchanged. The evaluation operator in this example is a fitness function to evaluate the fitness values f_i of these solutions. As for the determination operator of this example, the metaheuristic algorithm will employ a predefined mechanism to determine and choose solutions that have a better chance to end up being the optimal solution. As this example shows, because f_1 is better than the others, it has a better chance to be passed on to the next iteration.

B. Metaheuristic Algorithms for Scheduling

1) *Single-Solution-Based Metaheuristics:* A well known iterative greedy algorithm, hill climbing (HC), is given in Algorithm III-B to illustrate the basic structure and idea of metaheuristics. **T** and **E** on line 3 of Algorithm III-B indicates

[2]This operator is also known as the move, recombination, change, or alter operator in other research works.

Algorithm 2 Hill Climbing

```
1: Randomly create the initial solution s
2: While the termination criterion is not met
3:     v = NeighborSelection(s)          T→ E
4:     If v is better than s, then s = v      D
5: End
```

Algorithm 3 Simulated Annealing

```
1: Set the initial temperature according to the annealing
   schedule
2: Randomly create the initial solution s
3: While the termination criterion is not met
4:     v = NeighborSelection(s)          T→ E
5:     If v satisfies the probabilistic acceptance criterion
   D
6:     s = v
7: Update according to the annealing schedule          T
8: End
```

that this operation involves two mechanisms: transition and evaluation of HC. As far as this example is concerned, the transition mechanism is used to create neighbors on which the evaluation mechanism is performed and then the best solution is chosen. After that, **D** on line 4 denotes the determination mechanism of HC performed which compares the new solution (also called candidate solution) with the current solution and then uses the better one as the starting point of search at the next iteration.

The simulated algorithm (SA) can be used to illustrate that a simple change (as shown in lines 5 to 7 of Algorithm III-B1) can make an iterative greedy algorithm a metaheuristic which was first presented by Kirkpatrick *et al*. [28] and Černý *et al*. [29]. The basic idea of SA is to occasionally accept nonimproving solutions so as to escape from local optima during the convergence process. To emulate the annealing process, a method that is commonly used in computing the probability of accepting nonimproving solutions is defined as follows:

$$P_a = \exp\left(\frac{-f(v) - f(s)}{\Psi}\right) \tag{1}$$

where $f(\cdot)$ denotes the evaluation function; s the current solution; v the new solution; and Ψ the temperature. As shown in Algorithm III-B1, like the HC, the SA is an iterative algorithm that starts with a random initial solution and searches the neighbors of the current solution for the next solution; unlike the HC, bad solutions have a small chance to be accepted as the next search direction (solution) of SA. That is why SA usually can provide a better result than HC because by accepting bad solutions from time to time, it can mitigate the premature problem[3] of almost all iterative algorithms, such as local search algorithm, DAs, and stochastic algorithms.

Four different strategies—interchange neighborhood, shift neighborhood, ordered search, and random search—were com-

[3]By the premature problem, it means the problem of falling into local optimum at early iterations on the convergence process.

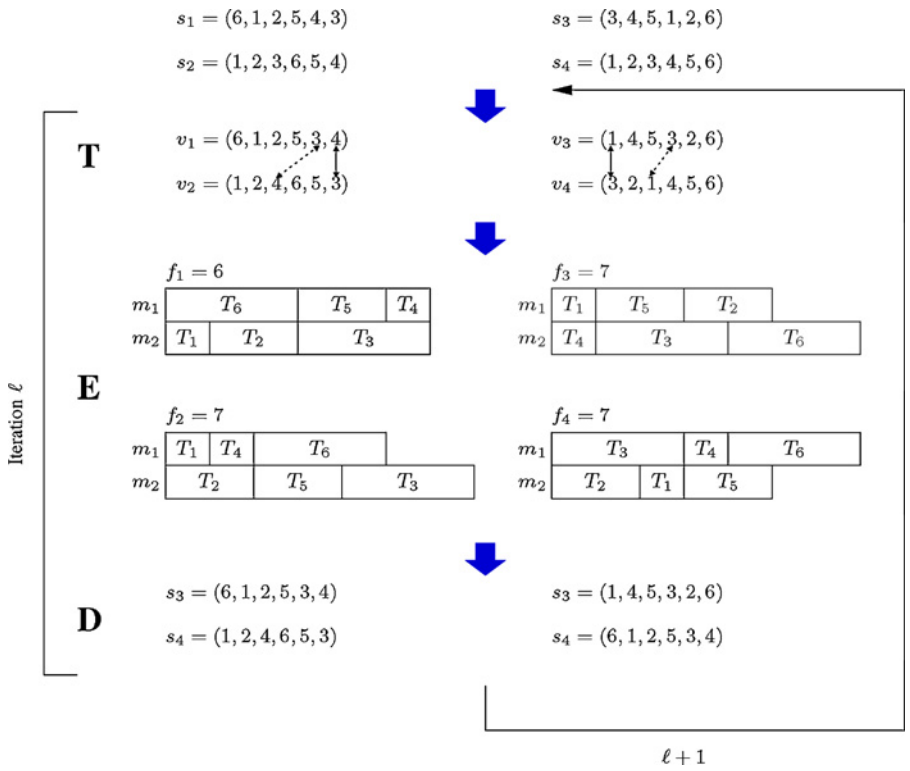

Fig. 3. Simple example illustrating how a metaheuristic algorithm for the scheduling problem works. **T**, **E**, and **D** denote, respectively, the transition operator, the evaluation operator, and the determination operator.

pared in [30] for analyzing what kind of combination can provide better results for the flow-shop scheduling problem than for the others. As observed by Osman and Potts [30], the SA with shift neighborhood and random search provides better results than the other combinations which also outperform Nawaz, Enscore, and Ham's algorithm (NEH) and its variants in terms of the quality of the solutions on average. A more recent study [17] applied SA to the job-shop scheduling problem in which the neighbor selection operator (i.e., the transition operator with the framework) only swaps (transits) subsolutions on the critical path. In addition, simulations are given to show the impact of annealing schedule.

Different from SA that accepts bad solutions occasionally to escape from the local optima, Glover [31]–[33] presented the tabu search (TS) to avoid searching the same solutions frequently. The TS keeps track of solutions recently visited into a tabu list to solve this problem, as shown in lines 1 and 4 of Algorithm III-B1. Like HC and SA, the TS algorithm starts with a single solution and then tries to find a neighbor of the current solution as the next solution, with the constraint that the new solution cannot be in the tabu list. If the new solution v (as shown in line 5 of Algorithm III-B1) is accepted, then it will be inserted into the tabu list and will stay in the tabu list until it is replaced by another new solution. The convergence process of TS can then avoid searching the same solutions for a while depending on the size of the tabu list, thus forcing the search algorithm to search for regions not on the same local optimum region.

The simple TS was employed to solve the flow-shop scheduling problem in [15]. In order to reduce the computation

Algorithm 4 Tabu Search

1:	Empty the tabu list	
2:	Randonly create the initial solution s	
3:	**While** the termination criterion is not met	
4:	v = NonTabu-NeighborSelection(s)	**T**→ **E**
5:	If v satisfies the improving conditions	**D**
6:	s = v	
7:	Update the tabu list based on s	**T**
8:	**End**	

time of TS, a parallel version of TS that uses the master and slave model was presented in the same study. The master is responsible for keeping the tabu list, while the slaves are responsible for finding better candidate solutions from the neighbors of the current solutions not in the tabu list. According to the observation of Taillard [15], TS can provide a better result than NEH in terms of not only the quality but also the computation time. In addition, the parallel version of the TS can be used to reduce the computation time of the TS. A more recent study [34] presented a new search strategy for TS for the job-shop scheduling problem. Two methods are employed in this paper to create the neighbors: the first method is to swap any two subsolutions (operations) on the same machine if they are on the critical path; the second method is to swap at least one subsolution not on the critical path, for if the adjustment is applied to only subsolutions on the critical path, makespan cannot be reduced. Moreover, the same study presented an adjustable tabu list the size of which can be dynamically increased and decreased to balance the

Algorithm 5 Genetic Algorithm

```
1: Randomly create the initial population s
2: While the termination criterion is not met
3:     f = FitnessFunction(s)          E
4:     v = Selection(s, f)             D
5:     v′ = Crossover(v)               T
6:     v″ = Mutation(v0)               T
7:     s = Reproduction(v00)
8: End
```

intensification and diversification of a search strategy. Methods for creating the neighbors, swapping, as well as insertion and block insertion were presented in [35]. In the same study, it was shown that the speed of the algorithm (BF-TS) they proposed is faster than T-TS [15] because BF-TS can reduce the candidate list from $(n-1)^2$ down to $2n$ and the number of iterations from 4000–50 000 to 100–500 while providing the similar result, on average.

2) *Population-Based Metaheuristics:* Two different characteristics of the population-based metaheuristics can be used to differentiate them from single-solution-based metaheuristics. First, the number of directions (solutions) searched each iteration is different. Second, the way the information searched (experience) propagated iteration by iteration is different, especially most population-based metaheuristics add other ways to exchange the search information at each iteration.

Pioneered by Holland [36], the genetic algorithm (GA) is definitely one of the most important population-based algorithms in terms of not only its performance but also the easiness of applying it to many problem domains. In terms of the report of Google Scholar [37], [36] was cited 14 712 times and [38] was cited 24 941 times. Both are listed in the top 100 cited works of Google Scholar (31 and 10, respectively). As shown in Algorithm III-B2, the GA contains the initialization, selection, reproduction, crossover, and mutation operators to mimic the process of natural evolution. Chromosomes (also called individuals) that represent the solutions are created randomly in the initialization stage. Unlike metaheuristic algorithms, which use the objective function to decide which solutions are better, the so-called fitness function is employed by GA to evaluate which solutions fit better. More important is that the fitness function can be useful to differentiate the solutions based on other criteria, such as rank or proportion. The design of the selection operator, which plays the role of determining the search directions at the next iteration, and the fitness function often goes hand by hand. The crossover and mutation operators are used to transit the solutions. More precisely, the crossover operator is aimed at exchanging the information between solutions, whereas the mutation operator is aimed at escaping from the local optimum.

Nine representations were pointed out in [39] for the job-shop scheduling problem. Generally speaking, most of these representations can also be applied to other metaheuristic scheduling problems. The classifier system to create schedule rules for the job-shop scheduling problem was presented in [18]. The GA was also used in a more recent study [40] to schedule the Hubble space telescope, at which the job ID and time segment number are used to encode the solutions. The same study further indicated that the random crossover operator provides better results than the other crossover operators, such as smart crossover and evolving crossover. Some encoding methods for the single machine, multiple machine, and dynamic scheduling problems were introduced in [41]. The GA was employed in [42] to solve the scheduling problem. Instead of using only task sequence to represent the schedule of all the tasks on a multiple machine, the order of the tasks to be executed in each processor is used to represent the schedule so that their transitions are all on the task graph to guarantee that the new solutions will be legal. A different research direction was described in [43] that applies multiobjective GA to the flow-shop scheduling problem to select the chromosomes on the convergence process. The fitness function is redesigned accordingly for the multiobjective GA, and the elite strategy is used to preserve better solutions to later generations. The simulation results described in [43] showed that the multiobjective GA provides better results than the single-objective GA. For the scheduling problem of heterogeneous computing environments, a directed acyclic graph (DAG) was used in [44] to represent the chromosomes (solutions) on the convergence process of the GA. Simulations given in the same study showed that the rank-based selection method is better than the value-based selection method.

In addition to the development of GA in the 1990s, swarm intelligence (SI) [45] is another promising approach developed at about the same time. One well known SI is ant colony optimization (ACO) pioneered by Dorigo and his colleagues [46]–[48]. As shown in Algorithm III-B2, like most SI, ACO is built on the social insect metaphor for solving optimization problems. An interesting characteristic of ACO is the pheromone table for recording the search experiences of all the ants from the initial stage up to the current iteration. ACO performs the transition, evaluation, and determination operations based on the values in the pheromone table (τ). This explains how all the ants of ACO share the search information. Another characteristic of ACO is that all the routing paths (solutions) are constructed step by step by all the ants which can be regarded as a constructive transition. Just like ants searching for routing paths for food, the solution construction operator of ACO plays the role of performing all the three operators: transition, evaluation, and determination. One of the well-known construction methods was presented in [46], which is defined as

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{j \in \mathcal{N}_i^k}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}, & \text{if } j \in \mathcal{N}_i^k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $\mathcal{N}_i^k$ denotes the set of candidate subsolutions, i.e., the subsolutions that have not visited by ant k yet and thus can be selected by ant k at subsolution i; τ_{ij} and η_{ij} denote, respectively, the pheromone value and the heuristic value associated with e_{ij}.

The earliest attempt to use ACO to solve the scheduling problem is the work described in [19] in which the ant system (AS) is employed to solve the job-shop problem. The results, however, are far from optimal. A more recent study [49]

Algorithm 6 Ant Colony Optimization

```
1: Initialize the pheromone values
2: While the termination criterion is not met
3:     v = SolutionConstruction(τ)          T→ E→ D
4:     τ = PheromoneUpdate(v)               T
5:     s = LocalSearch(v)                   T→ E→ D
6: End
```

Algorithm 7 Particle Swarm Optimization

```
1: Initialize the position and velocity of particles
2: While the termination criterion is not met
3:     VelocityUpdate(s)                    T
4:     v = PositionUpdate(s)                T
5:     LocalBestUpdate(v)                   E→ D
6:     GlobalBestUpdate(v)                  E→ D
7:     s = v
8: End
```

fine-tuned the parameter values of AS to improve the result of applying ACO to the scheduling problem. The way the probability of selecting edges is computed was modified in [50] so that bad solutions are accepted occasionally, just like the SA, to adjust the ratio of intensification and diversification in the search. Several strategies to enhance the performance of ACO for the scheduling problem were discussed in [51], namely elite strategy, 2-opt strategy, and two pheromone evaluation methods, which can indirectly influence the search ability of ACO, thus providing better results than SA and GA. The hybrid ACO presented in [52] combines the beam search with ACO to improve the result of applying ACO by itself to the open-shop scheduling problem.

Another well known SI, called particle swarm optimization (PSO), was presented in [53]. Algorithm III-B2 gives an outline of the PSO which uses individual trajectories, local best, and global best to guide the search, i.e., the transition of solutions. By using positions and velocities of particles, PSO [53]–[55] is usually suited for continuous optimization problems. Lines 3–6 of Algorithm III-B2 show, respectively, the velocity, position, global best, local best update mechanisms. All these update mechanisms affect the search directions of PSO at later iterations. The velocity v_i^l and position p_i^l are updated as follows:

$$v_i^{\ell+1} = \omega v_i^{\ell} + a_1\varphi_1(\mathrm{pb}_i^{\ell} - p_i^{\ell}) + a_2\varphi_2(\mathrm{gb}^{\ell} - p_i^{\ell}) \tag{3}$$

and

$$p_i^{\ell+1} = p_i^{\ell} + v_i^{\ell+1} \tag{4}$$

where the subscript i denotes the particle number; ℓ the iteration number; pb_i^t the personal best position of the ith particle up to iteration ℓ; gb^{ℓ} the global best position so far; ω an inertial weight; φ_1 and φ_2 two uniformly distributed random numbers used to determine the influence of pb_i and gb; and a_1 and a_2 two constant values denoting, respectively, the cognitive and social learning rate.

Because the characteristics of the original PSO are well suited for continuous optimization problems, the very first thing to apply PSO to combinatorial optimization problems is to solve the way solutions are encoded or represented. Permutation-based and priority-based (random key) representations were pointed out in [20] as the two common encoding methods of PSO for the resource-constrained project scheduling problem. Another representation of PSO was employed in [56] in which "job-to-position" is used to represent the solution of the flow-shop scheduling problem. Each solution is encoded as which job will be performed on which position, which can be easily transformed to a sequence of schedules. In addition, the length of each solution is simply the number of jobs multiplied by the number of positions. PSO searches for solutions by associating with each solution a probability set first and then modifying this probability set to change the search directions on its convergence process. A more recent study [57] presented a hybrid PSO to solve the job-shop problem. They not only modified representation, movement, and velocity of particles but also used the TS as the local search method for improving the quality of the solution. A modified discrete PSO that employs the nearest neighbor and NEH to create a better starting point for the flow-shop scheduling problem was presented in [58]. In the same study, variable neighborhood descent is used to improve the quality of the solution.

IV. Scheduling on Cloud

A. Scheduling Problems on Cloud

Scheduling on cloud can be considered as an epitome of the studies of cloud; thus, although many studies [10] attempted to give a precise and clear definition and description of scheduling on cloud, there still exist various definitions. A simple way to define the scheduling problem on cloud is usually represented as the DAG [59]–[64] $G(V, E)$ where V denotes the set of tasks and E the set of directed edges that represent the dependencies among tasks. The scheduling on cloud can be formulated as follows [64]:

$$\begin{aligned} &\text{minimize } f(s) = C_{\max}(s) + \sum_{i=1}^{n}\sum_{j=1}^{m}\mathrm{TC}_{ij} \\ &\text{subject to} C_{\max}(s) \le U(s), \\ &\qquad \mathrm{TC}(s) \le B(s), \end{aligned} \tag{5}$$

where $C(s)$ denotes the makespan of solution s; $\mathrm{TC}(s)$ the total cost of solution s, which can be a combination of the computation cost and the cost of transferring the incoming and outgoing data; TC_{ij} the cost of processing the ith task on the jth machine; $U(s)$ the number of tasks that does not meet the deadline; and $B(s)$ the number of tasks that does not meet the stipulated budget.

To formulate the scheduling problem, some researchers [65] considered adding a penalty factor to confine the search of solutions so as to avoid solutions containing too many tardy tasks. To measure the quality of the scheduling results, various methods are employed in these studies which can be used to

adapt the aforementioned definition to fit the situation a user of the scheduling algorithm may face. The average profit, average utilization, and average response rate [60] are employed to measure the performance of scheduling algorithms for cloud. Jiang *et al.* [62] tried to minimize the cost of a scheduling solution for cloud in terms of the computation cost, the communication cost, and the earliest start time. A more recent study [66] focused on the SLA-based resource conditions, such as throughput, latency, and cost of service agreement. Moreover, according to our observations, the issues of dependency of jobs and virtual machines should be part of the definition on scheduling. To reduce the development cost of applications for cloud environments, simulation platforms, such as Cloudsim [67] for cloud computing, have been presented which can also be used to design and develop better scheduling algorithms.

B. Metaheuristic Scheduling Algorithms

In this section, we turn our discussions to metaheuristic scheduling algorithms for cloud environments, especially from the following four perspectives: representation, transition, evaluation, and determination. Then, a more detailed analysis will be given to these metaheuristics.

1) *GA Scheduling Algorithms:* Before the scheduling procedure enters the main iterative process of the GA, just like the other metaheuristics, the GA has to initialize itself by constructing an initial solution set that may affect its convergence process and speed. A two-stage method (test and computing fitness stages) was used in [68] to create a good initial solution set of GA. Some heuristic scheduling algorithms (e.g., round-robin and min-min) are used to create the candidate initial solutions first, and then some suitable candidate solutions (in terms of the completion times of tasks and the communication costs between resources) are selected as the initial solutions after they pass some tests.

Five representations are commonly used in encoding the scheduling solutions of GA: 1) binary; 2) n-task sequence; 3) tree; 4) random key; and 5) $n \times m$ matrix. The binary encoding was used in [69] to associate the makespans and generations with chromosomes the length of which are $n \times n_j$ where n denotes the number of jobs and n_j the number of operations needed by job j. That is, each chromosome represents the order of all the operations of all the jobs. For instance, let us assume that there are two jobs each of which requires two operations. The solution $s = (0, 0, 1, 1)$ represents the order with which the operations are performed. In this case, the order is as follows: the first operation of job 0 is first performed, then the second operation of job 0, then the first operation of job 1, and so on. The n-task sequence was used in [68] to encode the solutions of the GA. A sequence (vector) $s = (s_1, s_2, \ldots, s_n)$ where s_i denotes to which available resource task i is assigned is employed to represent a solution. For instance, $s_1 = 2$ denotes that task 1 is assigned to the second resource (also called machine or node). To represent the mapping relationship between the virtual machines, Sawant [70] and Gu *et al.* [71] used used the tree structure to encode the scheduling solutions of the GA. The transition operator, of course, needs to be reconsidered to guarantee the legality of chromosomes. An interesting representation was described in [72], which uses the random key to maintain the feasibility (task priority) of chromosomes, though this kind of representation requires additional encoding and decoding methods for the other operators of the GA. The last representation is to use the matrix to represent the task model which includes the relationships between jobs and processors (also called machines or nodes). The predicted execution time model presented in [73] uses this representation to record the predicted execution time of task i on processor j.

The selection strategy of the GA usually can be regarded as the evaluation and determination operators of metaheuristic algorithms. Different assumptions may lead to different considerations in the design of the fitness functions. For example, the scheduling goal (fitness function of GA) is not only the traditional makespan but also the power consumption [74]. The cost of data transfer, communication, and computation as well as the profit have all been considered in the design of the fitness functions. To count the total constraint violation degree for composite services, the fitness function can also be designed based on the following two conditions: one is to compute the fitness value based on the proportion of the scheduling result, while the other is to assume a small value when some constrains are violated [72]. For the GA, in addition to the roulette-wheel selection, elitism selection, and tournament selection operators [71], [75], [76] that are employed for scheduling on cloud, a variety of selection operators have also been developed.

Most GAs use two transition operators (crossover and mutation) to exchange the information between solutions and to alter the solutions. Both one-point and two-point crossovers are used by the GA to exchange the information between chromosomes. The fact that cycle crossover is preferred to the one-point and two-point crossovers was pointed out in [75]. For a specific chromosome representation, the crossover operator usually needs to be redesigned. The orthogonal crossover and mutation operators to transit the chromosomes are designed for representing agent grid [77]. For the mutation operator, swapping the genes of a chromosome is commonly used in the GA. Also, for the tree representation [70], the crossover and mutation operators need to be redesigned or adopted to make them applicable. Instead of fixing the crossover rate, some researchers [76] used the fitness ratio to dynamically adjust the crossover rate between chromosomes to retain chromosomes with high fitness values. Two mutation strategies for GA were presented in [77]. One is to randomly select a task from a chromosome and then put it into another chromosome while the other is to exchange two tasks from two chromosomes. A comparison between different combinations of the crossover, partially mapped crossover (PMX), order crossover (OX), and cycle crossover (CX) operators as well as the mutation, swap, and insertion operators was given in [78]. Their simulation results show that the PMX crossover operator combined with the swap mutation operator is faster than the other combinations to find the same results.

Several research works have been focused on redesigning the search strategy of the GA to enhance its performance. An intuitive method is to use the parallel version of the GA because it has been proved that it can reduce the computation

time of the GA for a variety of traditional optimization problems. For this reason, in [79], a coarse-grained parallel GA was employed to solve the scheduling problem for cloud, which eventually provides a better result than the simple GA. Another way to employ high-performance version of the GA was described in [80] where the immune GA is employed to modify the search strategy of the GA to increase the diversity of chromosomes to further improve the final result of the GA. An efficient framework, called multiagent GA (MAGA), was used in [81] where a grid structure is used in representing the population. This approach, however, requires that chromosomes be exchanged with their neighbors instead of with all the chromosomes. Same as the original version of MAGA, the final result of [81] is also better than the simple GA. In [77], chromosomes were split into two pools (high and low) in terms of the average energy consumption, and then chromosomes are selected from these two pools to exchange the information. The cooperative coevolutionary GA (CCGA) was presented in [82] to solve the deadline-constrained scheduling problem. Different from the search strategies described previously, each gene can come from any number of chromosomes. In other words, although most GAs use two parents to create the offspring, CCGA uses one up to n chromosomes to create the offspring where n denotes the size of the population. To compensate for the lack of fine-tuning ability, the local search operator is often used to accelerate the convergence speed and to improve the final result of the GA [83].

Because many scheduling problems have more than one objective, finding a balanced result has become a promising research topic. Among others, Kessaci *et al.* [84] and Zhao *et al.* [85] treated the scheduling problem as a multiobjective optimization problem. A intuitive way to solve the scheduling problem is to employ the well-known multiobjective GA (NSGA II) the results of which take into account the cost of CPU, memory, and bandwidth at the same time [85]. Another study in using the multiobjective GA [84] took into consideration the energy consumption of data center, CO_2 emissions, and generated profits to reduce the utility rate of energy consumption.

2) *Ant Colony Optimization Scheduling Algorithms:* The basic idea of ACO is to use ant to construct the solution (routing path) step by step, by using the so-called pheromone matrix. Each ant is like living in a network environment because each city (also called node) can be regarded as a computer node in a network. For this reason, ACO is very suitable for emulating the status of real networks. To solve the scheduling problem on grid or cloud computing environments, several studies [86]–[88] used an n by m expected execution time (ET) matrix where the entries ET_{ij} denote the expected execution time of task T_i on machine M_j. The completion time (CT_{ij}) of ith task on the jth machine is defined as $\mathrm{CT}_{ij} = b_i + \mathrm{ET}_{ij}$, where b_i is the beginning time (ready time) of task T_i. Then, the makespan of a scheduling problem can be computed by using CT_{ij}. Most studies measure the workload of a task in terms of million instructions per second (MIPS) [87]. In addition to the makespan that is used to measure the throughput of the system, the flow time is often used to measure the QoS of the system [88]. The solution of ACO for a scheduling problem can be represented as composed of a set of subsolutions each of which indicates to which machine a task is assigned (scheduled) and is constructed with a probability

$$P_{ij} = \frac{\tau_{ij}\eta_{ij}(1/\mathrm{CT}_{ij})}{\sum \tau_{ij}\eta_{ij}(1/\mathrm{CT}_{ij})} \tag{6}$$

where τ_{ij} denotes the pheromone value associated with T_i and M_j; η_{ij} the heuristic information. The solution of ACO-based algorithms can be easily obtained by having them assign (or schedule) all the tasks to the machines.

By using ACO to solve the scheduling problems on a cloud environment, many computation resources can often be considered at the same time on the convergence process, such as CPU usage, memory, and network bandwidth [89]. In [90], four types of heuristic information are used: reliability of cloud service, response time of cloud service, cost of cloud service, and security ability of cloud service. Because the number of virtual machines (VM) represents the computing resources on a cloud environment, in [91], the number of processors on a VM, MIPS of each processor of a VM, communication bandwidth, average execution time of a VM, expected execution time of a task in a VM are all taken into account in the computation of the probability P_{ij} for constructing each subsolution of ACO. Still, another way to modify the pheromone update rule was described in [92] where additional pheromone is added to describe when a task is assigned to a machine. To improve the performance of ACO, the local search operator plays an important role. An intuitive local search method is to swap subsolutions (tasks) between machines [93]. As far as the search strategy is concerned, dividing ants into different types is another method to improve the end results of ACO. Two kinds of ants, red and black, were employed in [94] where the red ants are used for estimating the system resource, while the black ants are used for determining the resource allocation.

3) *PSO Scheduling Algorithms:* Similar to the GA for scheduling on cloud, which uses the DAG to describe the tasks in a workflow application, some studies also employed DAG for PSO [95]. Because the original design of PSO is for continuous problems, several methods have been developed to represent solutions for discrete problems (e.g., scheduling problem), such as random key representation, transformation, and priority-based representation. One of the key issues is that the transition operator needs to be redesigned to fit the requirements of its representation. Another key issue is how to encode the solution of PSO for scheduling. A common method is to encode in each particle a set of $\langle$task T_i, machine $M_j\rangle$ pairs where each pair denotes the assignment of task T_i to machine (service) M_j. Just like for ACO, which uses the ET matrix, for PSO, each particle also uses a n by m matrix to encode the solution. The only difference is that, instead of the expected execution time, each entry here assumes the probability p_{ij} of assigning task T_i to machine M_j [76], [96]. To keep more information about the network status, the fuzzy scheme [97] was employed for the PSO where the size of the fuzzy matrix is also n by m. To decode the fuzzy matrix, the maximum element of each column is selected.

Like the GA and ACO, modified fitness functions are also used to characterize the status of network on cloud. The total execution cost, communication cost, memory, and processing resource capacity of each machine are usually considered in PSO for scheduling on cloud [65]. Adding a penalty factor into the fitness function is a precise method to guide the search direction to avoid exceeding the system capacity. The main motivation for applying the mutation operator to PSO is to retain or increase the search diversity of each particle on the convergence process [98], [99]. Just like the mutation operator, the crossover operator has also been tried for improving the scheduling result of PSO [100]. From the perspective of search strategy, PSO resembles GA which suffers from the premature problem and lacks the fine-tuning ability. Integrating the local search operator into the search procedure of PSO is a useful way to fine-tune the scheduling results of PSO [99].

4) *Hybrid Metaheuristic Scheduling Algorithms:* The underlying idea of hybrid metaheuristics is to integrate two or more metaheuristic algorithms into a single algorithm to leverage the strengths of all the underlying algorithms so as to enhance the performance in terms of either the quality of the result or the computation time or both. Three different kinds of combinations are frequently used in hybrid metaheuristic scheduling algorithms: 1) combining population-based algorithm with single-solution-based algorithm; 2) combining two population-based algorithms; and 3) combining metaheuristic algorithm with other heuristic algorithm.

In the case of combining population-based algorithm with single-solution-based algorithm, the population-based algorithm is normally used to guide the global search, while the single-solution-based algorithm is employed to fine-tune the solution of the population-based algorithm. For instance, the hybrid metaheuristic scheduling algorithm described in [101] used SA as an operator of GA. To avoid getting stuck in a particular region, the TS is definitely one of the best choices. So the hybrid metaheuristic scheduling algorithm described in [100] used not only the crossover operator to exchange the information between particles of PSO but also TS to avoid searching the same regions for a while.

Because most population-based metaheuristic algorithms are capable of guiding the search directions toward the global optimum, combining two population-based metaheuristics should have good reasons. Otherwise, it may slow down the convergence speed without getting a better result. One of the most important reasons for combining two population-based metaheuristics is because of the difference in the convergence speed [102]. This implies that combining two population-based metaheuristics with different convergence characteristics may postpone the time of convergence. Another reason is that combining metaheuristic algorithms with different search strategies into a single metaheuristic algorithm may increase the chance to find a better result. From the viewpoint of information sharing, we combine two different population-based metaheuristics because different metaheuristics have different characteristics. If we can leverage the strengths of each algorithm into a single algorithm, we may be able to enhance its performance. A simple example is the algorithm described in [103] where two metaheuristic algorithms with different convergence characteristics are combined. More precisely, the algorithm integrates PSO and GA where the GA shares all the information between chromosomes (solutions), while PSO shares only the best solution with others.

Most hybrid algorithms utilize one metaheuristic algorithm and one nonmetaheuristic algorithm. Good examples are the dynamic cloud scheduling algorithm (DCSA) [104], which combines a support vector machine (SVM) with GA, and the adaptive hybrid heuristic (AHH) [64], which combines GA with dynamic critical path (DCP). To dynamically schedule the incoming tasks, a resource prediction mechanism was developed in the DCSA [104]. The system log data (CPU and memory usage) is first used to train the classification rules (classifier) of an SVM. Then, these rules can be used to predict resource availability on the convergence process of the GA. Because the DCSA uses system logs of nodes of real cloud systems, the search directions of the metaheuristic algorithm are generally closer to the status of the system. Like DCSA, AHH [64] was developed to dynamically schedule the tasks for services on cloud. AHH consists of two phases: GA and DCP. The first phase is to use the GA to find the scheduling solution that has a minimum execution cost in terms of the budget, deadline, and other requirements specified by the user. After that, the DCP is used to dynamically schedule the read tasks on the system.

C. Discussion

The scheduling for cloud computing environments has a relatively short history, but it is an important technology for modern computing systems. A number of important survey articles on scheduling for grid and cloud can be found in the literature [105]–[110]. In summary, most recent research works using metaheuristics are focused on three things: modifying the operators, modifying the fitness function, and hybrid metaheuristics.

- *Modifying the operators:* The main focus is on redesigning the transition operators or adding transition operators inspired by or of other scheduling algorithms. Both affect the search strategy of the original metaheuristic algorithm. In addition, some modifications may need to ensure that solutions created by the new transition operators are legal.
- *Modifying the fitness function:* The main focus is on redesigning or adding the fitness function to fit better with the cloud environment in question. As a consequence, the cost of data transfer, the cost of communication, the cost of computation, the profit, and even the energy consumption (e.g., CO_2 emissions) are continuously added to the fitness function. For some studies, the importance of each objective is no longer the same, that is, the relationships between these objectives have been redesigned. A good example is that cost is no longer considered independently but is considered relative to price.
- *Hybrid metaheuristics:* Different from modifying the transition operators or adding the local search operator, the basic idea of hybrid metaheuristics is to use other scheduling algorithms or domain knowledge to enhance the performance of the original algorithm, i.e., use the

strength of other scheduling algorithms to compensate for the weakness of the original metaheuristic algorithm. A good example is a combination of GA and SA where the GA plays the role of finding the global search directions, while the SA plays the role of fine-tuning the solutions found by GA.

In summary, because using metaheuristic algorithms to solve the scheduling problems on cloud is still at its infancy, numerous studies in the literature are simply not matured enough in terms of not only the description and definition of the problems but also the fitness of the operators of the metaheuristics for scheduling on cloud. As a result, most of the proposed algorithms are subject to improvement. This is not saying that these immature studies are not important. Instead, they are eventually providing not only useful experience but also foundations to advance later research works in this area. The main advantages of using metaheuristic algorithm for scheduling are similar to those of using them for traditional optimization problems: 1) metaheuristics generally provide better results than DAs in terms of the quality; and 2) metaheuristics generally find approximate solutions faster than traditional exhaustive algorithms in terms of the computation time. These characteristics imply that the metaheuristic algorithms provide a more flexible way to find solutions of scheduling problems. On the contrary, the disadvantages of metaheuristic scheduling are 1) metaheuristics generally are slower than DAs; and 2) solutions are not guaranteed to be optimal. That is why the recent trend in the other research areas is to preprocess the input data so as to accelerate the execution time of metaheuristic algorithms.

V. Conclusion

Using metaheuristics to solve the scheduling problem is like putting a robot in a big maze. How fast the robot finds the exit depends to a large extent on its vision (search ability) and intelligence (decision ability). Two of the most important abilities these research works need to consider are intensification and diversification. As the names suggest, an intensified search implies searching a small region intensively to find the best solution in that region, called local optimum. In contrast, a diversified search means searching a larger region for a solution that is better than the local optimum. Unfortunately, a larger region does not always guarantee a better solution. These observations tell us that it is important to seek a balance between intensification and diversification, which may have a strong impact on the quality of the scheduling results. The focus of this paper was on a survey of metaheuristic algorithms for scheduling on cloud. To make it easier for the audience of this paper to fully understand all the metaheuristic algorithms described herein, all the metaheuristic algorithms were placed in a unified framework.

Nowadays, because cloud computing is relatively new and nothing is really clear, the assumptions, objectives, and limitations of scheduling problems on cloud differ from study to study, meaning that although scheduling problems are not new at all, studies on scheduling problems on cloud are still at their infancy. A more applicable problem definition is needed in the future research on cloud scheduling. Although its definition is not clear now, a better scheduling method definitely has a higher potential to enhance the performance of most parts of a cloud computing system. A simple way to develop the scheduling algorithm is to consider the level of services such as software as a service, platform as a service, or infrastructure as a service. Just like most research subjects, scheduling on cloud computing also fits the product life cycle theory that is expected to undergo the introduction, growth, maturity, saturation, and decline stages. Scheduling on cloud computing today is somewhere between the introduction and growth stages. Managing the Internet of Things devices and multimedia contents and conserving energy more effectively are some of the important research trends on future cloud computing scheduling.

References

[1] M. Pinedo, *Scheduling: Theory, Algorithms and Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, Inc., 1995.

[2] P. Chrétienne, E. G. Coffman, J. K. Lenstra, and Z. Liu, Eds., *Scheduling Theory and Its Applications*. New York: Wiley, 1995.

[3] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, 2008.

[4] C. N. Potts and V. A. Strusevich, "Fifty years of scheduling: A survey of milestones," *J. Oper. Res. Soc.*, vol. 60, no. S1, pp. S41–S68, 2009.

[5] R. Ramasesh, "Dynamic job shop scheduling: A survey of simulation research," *Omega*, vol. 18, no. 1, pp. 43–57, 1990.

[6] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Res. Logistics Quart.*, vol. 1, no. 1, pp. 61–68, 1954.

[7] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generat. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[8] G. F. Pfister, *In Search of Clusters*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[9] I. Foster, "The grid: A new infrastructure for 21st century science," in *Grid Computing: Making the Global Infrastructure a Reality*. New York: Wiley, 2003, pp. 51–63.

[10] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. Grid Comput. Workshop*, 2008, pp. 1–10.

[11] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. Int. Joint Conf. INC, IMS IDC*, 2009, pp. 44–51.

[12] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," Dept. Elect. Eng. Comput. Sci., Univ. Calif., Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[13] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[14] M. R. Garey and D. S. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.

[15] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," *Eur. J. Oper. Res.*, vol. 47, no. 1, pp. 65–74, 1990.

[16] T. E. Morton and D. W. Pentico, *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management* (Wiley Series in Engineering and Technology Management). New York: Wiley, 1993.

[17] P. J. M. van Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Oper. Res.*, vol. 40, no. 1, pp. 113–125, 1992.

[18] M. R. Hilliard, G. E. Liepins, and M. Palmer, "Machine learning applications to job shop scheduling," in *Proc. Int. Conf. Ind. Eng. Appl. Artif. Intell. Expert Systems*, 1988, vol. 2. pp. 728–737.

[19] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, "Ant System for Job-shop Scheduling," *Belgian J. Oper. Res., Statist. Comput. Sci.*, vol. 34, no. 1, pp. 39–53, 1994.

[20] H. Zhang, X. Li, H. Li, and F. Huang, "Particle swarm optimization-based schemes for resource-constrained project scheduling," *Autom. Construct.*, vol. 14, no. 3, pp. 393–404, 2005.

[21] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt, and J. Węglarz, *Scheduling Computer and Manufacturing Processes*. New York: Springer-Verlag, 2001.

[22] Y. T. J. Leung, *Handbook of Scheduling: Algorithms, Models and Performance Analysis*. London, U.K.: Chapman & Hall, 2004.

[23] K. R. Baker, *Introduction to Sequencing and Scheduling*. New York: Wiley, 1974.

[24] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Ann. Discr. Math.*, vol. 5, pp. 287–326, Jan. 1979.

[25] P. P. Wang, "Static and dynamic scheduling of customer arrivals to a single-server system," *Naval Res. Logistics*, vol. 40, no. 3, pp. 345–360, 1993.

[26] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer-Verlag, 1996.

[27] H. H. Hoos and T. Stützle, *Stochastic Local Search: Foundations & Applications*. Amsterdam, The Netherlands: Elsevier, 2004.

[28] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[29] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *J. Optimization Theory Appl.*, vol. 45, no. 1, pp. 41–51, 1985.

[30] I. H. Osman and C. N. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989.

[31] F. Glover, "Tabu search—part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.

[32] F. Glover, "Tabu search—part II," *ORSA J. Comput.*, vol. 2, no. 1, pp. 4–32, 1990.

[33] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer, 1997.

[34] M. Dell'Amico and M. Trubian, "Applying tabu search to the job-shop scheduling problem," *Ann. Oper. Res.*, vol. 41, nos. 1–4, pp. 231–252, 1993.

[35] M. Ben-Daya and M. Al-Fawzan, "A tabu search approach for the flow shop scheduling problem," *Eur. J. Oper. Res.h*, vol. 109, no. 1, pp. 88–95, 1998.

[36] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. of Michigan Press, 1975.

[37] Top 100 Cited Works of Google Scholar. (2008, Jul. 21) [Online]. Available: http://www.cse.wustl.edu/ loui/goocites.html

[38] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.

[39] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms—I: representation," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 983–997, 1996.

[40] J. L. Sponsler, "Genetic algorithms applied to the scheduling of the hubble space telescope," *Telemat. Informat.*, vol. 6, nos. 3–4, pp. 181–190, 1989.

[41] J. E. Biegel and J. J. Davern, "Genetic algorithm and job shop scheduling," *Comput. Ind. Eng.*, vol. 19, nos. 1–4, pp. 81–91, 1990.

[42] E. S. H. Hou, N. Ansari, and H. Ren, "A genetic algorithm for multiprocessor scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 5, no. 2, pp. 113–120, Feb. 1994.

[43] T. Murata, H. Ishibuchi, and H. Tanaka, "Multi-objective genetic algorithm and its applications to flowshop scheduling," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 957–968, 1996.

[44] L. Wang, H. J. Siegel, V. R. Roychowdhury, and A. A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach," *J. Parallel Distrib. Comput.*, vol. 47, no. 1, pp. 8–22, 1997.

[45] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. New York: Wiley, 2006.

[46] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[47] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[48] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, July 2004.

[49] S. van der Zwaan and C. Marques, "Ant colony optimisation for job shop scheduling," in *Proc. Workshop Genetic Algorithms Artif. Life*, 1999, pp. 1–8.

[50] V. T'kindt, N. Monmarché, F. Tercinet, and D. Laügt, "An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 142, no. 2, pp. 250–257, 2002.

[51] D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 333–346, Aug. 2002.

[52] C. Blum, "Beam-ACO: Hybridizing ant colony optimization with beam search: An application to open shop scheduling," *Comput. Oper. Res.*, vol. 32, no. 6, pp. 1565–1591, 2005.

[53] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Nov.–Dec. 1995, pp. 1942–1948.

[54] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*. Berlin, Germany: Springer-Verlag, 1998, pp. 591–600.

[55] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[56] C. J. Liao, C. T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3099–3111, 2007.

[57] D. Y. Shaa and C. Y. Hsu, "A hybrid particle swarm optimization for job shop scheduling problem," *Comput. Ind. Eng.*, vol. 51, no. 4, pp. 791–808, 2006.

[58] Q. K. Pan, M. Fatih Tasgetiren, and Y. C. Liang, "A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2807–2839, 2008.

[59] J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on IaaS cloud systems," *J. Parallel Distrib. Comput.*, vol. 72, no. 5, pp. 666–677, 2012.

[60] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou, "Profit-driven scheduling for cloud services with data access awareness," *J. Parallel Distrib. Comput.*, vol. 72, no. 4, pp. 591–602, 2012.

[61] B. Saovapakhiran, G. Michailidis, and M. Devetsikiotis, "Aggregated-dag scheduling for job flow maximization in heterogeneous cloud computing," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2011, pp. 1–6.

[62] H.-J. Jiang, K.-C. Huang, H.-Y. Chang, D.-y. Gu, and P.-J. Shih, "Scheduling concurrent workflows in HPC cloud through exploiting schedule gaps," in *Algorithms and Architectures for Parallel Processing*, vol. 7016. Berlin, Germany: Springer, 2011, pp. 282–293.

[63] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *J. Grid Comput.*, vol. 3, nos. 3–4, pp. 171–200, 2005.

[64] M. Rahman, X. Li, and H. Palit, "Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops*, May 2011, pp. 966–974.

[65] P. Y. Yin, S. S. Yu, P. P. Wang, and Y. T. Wang, "A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems," *Comput. Standards & Interfaces*, vol. 28, no. 4, pp. 441–450, 2006.

[66] Q. Li, "An optimal algorithm for resource scheduling in cloud computing," in *Advances in Multimedia, Software Engineering and Computing*, vol. 129. Berlin, Germany: Springer, 2012, pp. 293–299.

[67] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw.: Practice Exp.*, vol. 41, no. 1, pp. 23–50, 2011.

[68] A. G. Delavar and Y. Aryan, "A synthetic heuristic algorithm for independent task scheduling in cloud systems," *Int. J. Comput. Sci. Issues*, vol. 8, no. 6, pp. 289–295, 2011.

[69] D. W. Huang and J. Lin, "Scaling populations of a genetic algorithm for job shop scheduling problems using MapReduce," in *Proc. IEEE 2nd Int. Conf. Cloud Comput. Technol. Sci.*, Nov.–Dec. 2010, pp. 780–785.

[70] S. Sawant, "A genetic algorithm scheduling approach for virtual machine resources in a cloud computing environment," M.S. thesis, Dept. Comput. Sci., San Jose State University, San Jose, CA, USA, 2011.

[71] J. Gu, J. Hu, T. Zhao, and G. Sun, "A new resource scheduling strategy based on genetic algorithm in cloud computing environment," *J. Comput.*, vol. 7, no. 1, pp. 42–52, 2012.
[72] L. Ai, M. Tang, and C. J. Fidge, "QoS-oriented resource allocation and scheduling of multiple composite web services in a hybrid cloud using a random-key genetic algorithm," *Australian J. Intell. Inf. Process. Syst.*, vol. 12, no. 1, pp. 29–34, 2010.
[73] S. Tayal, "Tasks scheduling optimization for the cloud computing systems," *Int. J. Adv. Eng. Sci. Technol.*, vol. 5, no. 2, pp. 111–115, 2011.
[74] M.-S. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, A. Y. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1497–1508, 2011.
[75] E. M. Mocanu, M. Florea, M. I. Andreica, and N. Ţuapuş, "Cloud computing—task scheduling based on genetic algorithms," in *Proc. IEEE Int. Syst. Conf.*, Mar. 2012, pp. 1–6.
[76] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent tasks scheduling based on genetic algorithm in cloud computing," in *Proc. Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Sep. 2009, pp. 5548–5551.
[77] G. Shen and Y. Q. Zhang, "A shadow price guided genetic algorithm for energy aware task scheduling on cloud computers," in *Advances in Swarm Intelligence*, vol. 6728. Berlin, Germany: Springer, 2011, pp. 522–529.
[78] D. Dutta and R. C. Joshi, "A genetic algorithm approach to cost-based multi-QoS job scheduling in cloud computing environment," in *Proc. Int. Conf. Workshop Emerging Trends Technol.*, 2011, pp. 422–427.
[79] Z. Zheng, R. Wang, H. Zhong, and X. Zhang, "An approach for cloud resource scheduling based on parallel genetic algorithm," in *Proc. Int. Conf. Comput. Res. Develop.*, 2011, pp. 444–447.
[80] Y. Laili, L. Zhang, and F. Tao, "Energy adaptive immune genetic algorithm for collaborative design task scheduling in cloud manufacturing system," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage.*, Dec. 2011, pp. 1912–1916.
[81] K. Zhu, H. Song, L. Liu, J. Gao, and G. Cheng, "Hybrid genetic algorithm for cloud computing applications," in *Proc. IEEE Asia-Pacific Serv. Comput. Conf.*, Dec. 2011, pp. 182–187.
[82] L. Ai, M. Tang, and C. J. Fidge, "Resource allocation and scheduling of multiple composite web services in cloud computing using cooperative coevolution genetic algorithm," in *Proc. 18th Int. Conf. Neural Inf. Process.*, 2011, pp. 258–267.
[83] X. Wang, Y. Wang, and H. Zhu, "Energy-efficient multi-job scheduling model for cloud computing and its genetic algorithm," *Math. Probl. Eng.*, vol. 2012, pp. 589243-1–589243-16, Apr. 2012.
[84] Y. Kessaci, N. Melab, and E.-G. Talbi, "A Pareto-based GA for scheduling HPC applications on distributed cloud infrastructures," in *Proc. Int. Conf. High Perform. Comput. Simul.*, Jul. 2011, pp. 456–462.
[85] J. Zhao, W. Zeng, M. Liu, G. Li, and M. Liu, "Multi-objective optimization model of virtual resources scheduling under cloud computing and it's solution," in *Proc. Int. Conf. Cloud Serv. Comput.*, Dec. 2011, pp. 185–190.
[86] S. Fidanova and M. K. Durchova, "Ant algorithm for grid scheduling problem," in *Proc. Int. Conf. Large-Scale Sci. Comput.*, 2005, pp. 405–412.
[87] K. Kousalya and P. Balasubramanie, "Ant algorithm for grid scheduling powered by local search," *Int. J. Open Probl. Comput. Sci. Math.*, vol. 1, no. 3, pp. 222–240, 2008.
[88] D. Maruthanayagam and R. UmaRani, "Enhanced ant colony algorithm for grid scheduling," *Int. J. Comput. Technol. Appl.*, vol. 1, no. 1, pp. 43–53, 2010.
[89] X. Lu and Z. Gu, "A load-adapative cloud resource scheduling model based on ant colony algorithm," in *Proc. IEEE Int. Conf. Cloud Comput. Intell. Syst.*, Sep. 2011, pp. 296–300.
[90] H. Liu, D. Xu, and H. Miao, "Ant colony optimization based service flow scheduling with various QoS requirements in cloud computing," in *Proc. 1st ACIS Int. Symp. Softw. Netw. Eng.*, Dec. 2011, pp. 53–58.
[91] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *Proc. 6th Annu. ChinaGrid Conf.*, Aug. 2011, pp. 3–9.
[92] P. Mathiyalagan, S. Suriya, and S. N. Sivan, "Modified ant colony algorithm for grid scheduling," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 2, pp. 132–139, 2010.
[93] K. Kousalya and P. Balasubramanie, "To improve ant algorithm's grid scheduling using local search," *Int. J. Comput. Cognit.*, vol. 2, no. 2, pp. 71–79, 2009.
[94] A. Kant, A. Sharma, S. Agarwal, and S. Chandra, "An ACO approach to job scheduling in grid environment," in *Swarm, Evolutionary, and Memetic Computing*, vol. 6466. Berlin, Germany: Springer, 2010, pp. 286–295.
[95] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proc. IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 400–407.
[96] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in *Proc. Int. Conf. Comput. Intell. Security*, Dec. 2010, pp. 184–188.
[97] H. Liu, A. Abraham, and A. E. Hassanien, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm," *Future Generat. Comput. Syst.*, vol. 26, no. 8, pp. 1336–1343, 2010.
[98] B. Zarei, R. Ghanbarzadeh, P. Khodabande, and H. Toofani, "MHPSO: A new method to enhance the particle swarm optimizer," in *Proc. Int. Conf. Digital Inf. Manage.*, Sep. 2011, pp. 305–309.
[99] L. Guo, S. Zhao, S. Shen, and C. Jiang, "Task scheduling optimization in cloud computing based on heuristic algorithm," *J. Netw.*, vol. 7, no. 3, pp. 547–553, 2012.
[100] Z. Wang, K. Shuang, L. Yang, and F. Yang, "Energy-aware and revenue-enhancing combinatorial scheduling in virtualized of cloud datacenter," *J. Convergence Inf. Technol.*, vol. 7, no. 8, pp. 62–70, 2012.
[101] G. N. Gan, T. L. Huang, and S. Gao, "Genetic simulated annealing algorithm for task scheduling based on cloud computing environment," in *Proc. Int. Conf. Intell. Comput. Integr. Syst.*, Oct. 2010, pp. 60–63.
[102] Z. Yan-hua, F. Lei, and Y. Zhi, "Optimization of cloud database route scheduling based on combination of genetic algorithm and ant colony algorithm," *Procedia Eng.*, vol. 15, pp. 3341–3345, 2011.
[103] G. S. Sadasivam and D. Selvaraj, "A novel parallel hybrid PSO-GA using MapReduce to schedule jobs in Hadoop data grids," in *Proc. World Congr. Nature Biologically Inspired Comput.*, Dec. 2010, pp. 377–382.
[104] X. Shi and Y. Zhao, "Dynamic resource scheduling and workflow management in cloud computing," in *Proc. Int. Conf. Web Inf. Syst. Eng.*, 2011, pp. 440–448.
[105] A. Bala and I. Chana, "A survey of various workflow scheduling algorithms in cloud environment," in *Proc. Nat. Conf. Inf. Commun. Technol.*, 2011, pp. 26–30.
[106] N. Kaur, T. S. Aulakh, and R. S. Cheema, "Comparison of workflow scheduling algorithms in cloud computing," *Int. J. Adv. Comput. Sci. Appl.*, vol. 2, no. 10, pp. 81–86, 2011.
[107] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surveys*, vol. 31, no. 4, pp. 406–471, 1999.
[108] S. Pandey, "Scheduling and management of data intensive application workflows in grid and cloud computing environments," Ph.D. dissertation, Dept. Comput. Sci. Softw. Eng., Univ. Melbourne, Melbourne, Australia, 2010.
[109] A. Fida, "Workflow scheduling for service oriented cloud computing," M.S. thesis, Dept. Comput. Sci., Univ. Saskatchewan, Saskatoon, Canada, 2011.
[110] K. Rameshkumar and D. G. Amalarethinam, "Applying non-traditional optimization techniques to task scheduling in grid computing—an overview," *Int. J. Res. Rev. Comput. Sci.*, vol. 1, no. 4, pp. 33–38, 2010.

Chun-Wei Tsai received the M.S. degree in management information systems from the National Pingtung University of Science and Technology, Pingtung, Taiwan, in 2002, and the Ph.D. degree in computer science from National Sun Yat-sen University, Kaohsiung, Taiwan, in 2009.

He was a Post-Doctoral Fellow with the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, before joining the faculty of Applied Geoinformatics and then the faculty of Applied Informatics and Multimedia, Chia Nan University of Pharmacy & Science, Tainan, in 2010 and 2012, respectively, where he is currently an Assistant Professor. His current research interests include information retrieval, evolutionary computation, internet technology, and combinatorial optimization.

Joel J. P. C. Rodrigues (S'01–M'06–SM'06) received a five-year B.S. degree (licentiate) in informatics engineering from the University of Coimbra, Coimbra, Portugal, and the M.Sc. and Ph.D. degrees in informatics engineering from the University of Beira Interior, Covilhã, Portugal.

He is currently a Professor at the Instituto de Telecomunicações, University of Beira Interior, and a Researcher at the Instituto de Telecomunicações, Associated Lab, Portugal. He has authored or coauthored more than 170 technical papers in refereed international journals and conferences, book chapters, a book, and a patent. His current research interests include delay-tolerant networks, sensor networks, high-speed networks, eLearning, e-Health, and mobile and ubiquitous computing.

Dr. Rodrigues is the leader of the NetGNA Research Group (http://netgna.it.ubi.pt) and the founder and leader of the IEEE Communications Society's (ComSoc) Communications Systems Integration and Modeling Special Interest Group on modeling and simulation tools (http://mst.it.ubi.pt). He is the Vice-Chair of the IEEE ComSoc Technical Committee on eHealth and the Chair of the IEEE ComSoc Technical Committee on Communications Software. He has authored or coauthored over 250 technical papers in refereed international journals and conferences, book chapters, a book, and two patents. He is the Editor-in-Chief of the *International Journal on E-Health and Medical Communications* and served several special issues as a Guest Editor such as the IEEE TRANSACTIONS ON MULTIMEDIA, the Elsevier *Journal of Network and Computer Applications*, IET Communications, and *Journal of Communications*. He has served as the General Chair, Technical Program Committee (TPC) Chair, and Symposium Chair for many international conferences, including the IEEE International Conference on Communications/Global Communications Conference, the International Workshop on Computer-Aided Modeling Analysis and Design of Communication Links and Networks, the IEEE International Conference on Systems, Man, and Cybernetics, the International Conference on ITS Telecommunications, the International Conference on Networking and Computing, International Conference on Software, Telecommunications and Computer Networks, among others. He participated in tens of international TPCs and several editorial review boards including the IEEE Communications Magazine and the International Journal of Communications Systems. He is a licensed Professional Engineer, and a member of ACM SIGCOMM, a member of the Internet Society, an International Academy, Research, and Industry Association Fellow, and a senior member of the IEEE Computer Society, the IEEE Communications Society, and the IEEE Education Society.