ISSN : 0256-4602


The Institution of Electronics and Telecommunication Engineers

# IETE Technical Review

*Volume 28 • No. 2 • Mar-Apr 2011*

www.ietejournals.org

# Grid Task Scheduling: Algorithm Review

Tinghuai Ma, Qiaoqiao Yan, Wenjie Liu, Donghai Guan[1] and Sungyoung Lee[1]

School of Computer and Software, Nanjing University of Information Science and Technology, People's Republic of China,
[1]Department of Computer, Keyung Hee University, South Korea

## Abstract

As a new distributed heterogeneous computing platform, grid aims at achieving Internet-wide resource sharing and collaborative computing. Grid task scheduling (GTS) is the key issue of grid computing, and its algorithm has a direct effect on the performance of the whole system. In this paper, two key entities in GTS, applications and target systems, are defined first. And then two types of the most popular GTS algorithms, namely, meta-task GTS algorithm and directed acyclic graph GTS algorithm, are discussed in details in accordance with the classification of the traditional deterministic algorithm and heuristic intelligent algorithm. In addition, the comparative analysis is made among them. Finally, some main research directions of GTS are pointed out.

**Keywords**

*Directed acyclic graph, Grid computing, Grid task scheduling, Heuristic algorithms, Meta-task, Research topic, Swarm intelligent algorithms.*

## 1. Introduction

As a new distributed heterogeneous computing (HC) platform, grid computing aims at achieving Internet-wide resource sharing and collaborative computing. Task management, task scheduling, and resource management are the three key issues of grid computing [1]. In particular, grid task scheduling (GTS) plays an important role in the whole system, and its algorithms have a direct effect on the grid system performance. Task scheduling in HC environment has proved to be a NP-complete problem [2].

In order to solve this problem, various kinds of algorithms have been proposed since the grid system emerged. Generally, grid applications can be classified into meta-task applications and directed acyclic graph (DAG) applications, which will be discussed in Section 2. Based on this point, we divide the existing GTS algorithms into two types, namely, meta-task GTS algorithms and DAG GTS algorithms. Furthermore, they are classified into traditional deterministic algorithm and heuristic intelligent algorithm for their use of different optimization technologies. As been well known, task scheduling strategies usually include two types: static and dynamic strategies. The static strategy needs to accurately obtain some correlative information in advance, whereas the dynamic strategy assigns tasks according to the information collected at the runtime. So the former has the advantages of simple implementation, low running load, high efficiency, etc., and the latter has many uncertain factors and a relatively high scheduling overload. However, it can flexibly assign tasks during program execution and be well adapted to the different environments. As you might imagine, both of these two strategies are involved in GTS. And they are often used simultaneously in the algorithm design. That's to say, some algorithms use both static and dynamic strategies. For this reason, in our later classified discussion, we will not distinguish whether the algorithms is static or dynamic.

The remainder of this paper is organized as follows: Section 2 states the basic principles of GTS including grid applications and target systems. In Section 3, we provide a detailed classified discussion on the various GTS algorithms according to the type of grid applications. Several main researching directions on GTS are summed up and analyzed in Section 4. Finally, conclusions are drawn in Section 5.

## 2. Principles and Definition

GTS is to assign tasks to suitable processing units to execute under restraint by certain rules. It aims to maximize system throughput, improve resource utilization ratio, shorten task execute time as well as satisfy users' requirements of quality of service (QoS). Specifically, the objectives of GTS include optimal makespan, load balancing, QoS, economic principles, etc. [1]. The two key entities in GTS are applications and target systems: the former can be viewed as tasks in abstract; the latter can be described as a processing unit (PU) network. Correspondingly, task scheduling algorithm is the communication bridge between these two entities.

### 2.1 Grid Application

Conclusively, grid applications [3] can be described by a four-tuple $(T, \prec, C, D)$, where $T = (t_1, t_2, \cdots, t_n)$ represents the executable tasks set; $\prec$ denotes the partial ordering relation over $T$ tasks set, which are used to express the priority constraint relation between task, for example, if existing $t_i < t_j$, then task $t_i$ must be executed before task $t_j$; $C$ is a $n$-dimensional vector, let $C_i > 0 (1 \le i \le n)$ denote the amount of calculation of task $t_i$; $D$ is a $n \times n$ communication matrix, let $D_{i,j} \ge 0$ be the data quantity needing to be transferred from task $t_i$ to $t_j$.

In substance, grid applications can be classified into two categories as follows:
*(1) Meta-task applications:* Such applications can be simplified as a two-tuple (*T,C*). There are no priority constraint relation and data communication between tasks. That is, all the tasks can run independently.

*(2) DAG applications:* These applications are a typical model of $(T, \prec, C, D)$, which is usually described by DAG. A typical DAG task graph is illustrated in Figure 1. In the DAG graph, nodes represent tasks, directed edges denote the partial ordering relation between tasks, the node weights and edge weights are the amount of calculation and communication traffic of tasks, respectively.

### 2.2 Grid Target System

Grid target system is a PU network with a certain topological structure [4]. Each PU has its own processor and local restore unit. That is, PUs couldn't share memory, and they communicate with each other based on a message-driven mechanism.

A PU network can be described by an undirected graph $P = (R_p, E_p, C, D)$, where $R_p$ is the set of nodes that represent processors; $E_p$ is the set of edges that represent the communication link between processors; $C(r_i)$ denotes the calculation amount of processor $r_i$ per unit time; $D(r_i, r_j)$ denotes the communication traffic that communication link $(r_i, r_j) \in E_p$ can transfer per unit time. An example of the PU network is showed in Figure 2.

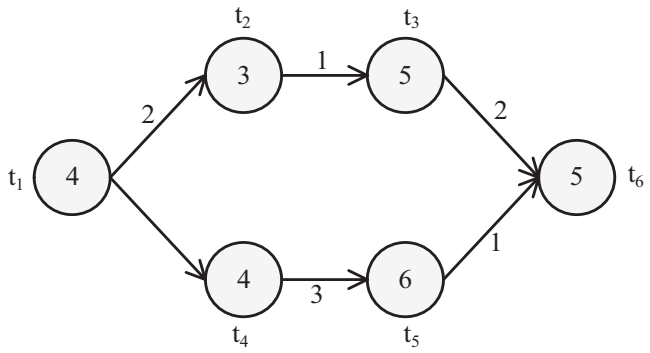



**Figure 1:** A DAG task graph.

## 3. GTS Algorithms

### 3.1 Meta-task GTS Algorithms

In this section, some typical meta-task GTS algorithms are introduced including traditional GTS algorithms and heuristic intelligent algorithms. The former addresses some deterministic algorithms, whereas the latter concentrates on some randomized intelligent algorithms.

#### *3.1.1 Traditional Deterministic Meta-task GTS Algorithms*

Early in 1999, Maheswaran *et al*. [5] studied dynamic mapping heuristics for a class of independent tasks used in the distributed HC systems, including minimum completion time (MCT), minimum execution time (MET), switching algorithm (SA), K-percent best (KPB), opportunistic load balancing (OLB), MinMin, MaxMin, and suffrage heuristic (SH). Ritchie *et al*. first reviewed several previous static scheduling algorithms including OLB, MET, MCT, MinMin, and MinMax [6]. Then combined with local search (LS), two new improved algorithms, MinMin plus LS and SH plus LS, were introduced. In 2004, Fujimoto *et al*. [7] conducted a comparison among round robin (RR) and five related algorithms including dynamic fastest processor to largest task first (DFPLTF), suffrage-C, MinMin, MaxMin, and work queue (WQ). Kim *et al*. [8] proposed a new online heuristic scheduling algorithm MECT. Compared with MET, MCT, and KPB, MECT can achieve better performance. In 2007, Luo *et al*. [9] investigated fast greedy heuristics for mapping a class of independent tasks onto HC systems, in which a collection of 20 greedy heuristics were implemented, analyzed, and systematically compared within a uniform model of task execution time. In 2009, Tseng *et al*. [10] proposed a novel GTS algorithm called ATCS-MCT taking execution time, weight, due date, and communication time factors

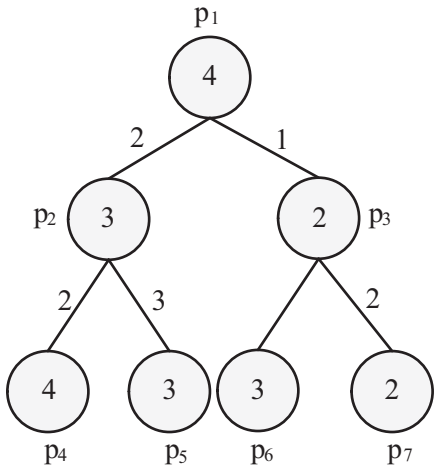



**Figure 2:** A processing unit graph.

together into account. Their experiments showed that ATCS-MCT not only achieved better makespan than MinMin did but also reduced costs.

In summary, traditional GTS algorithms mainly include OLB, MET, MCT, MinMin, MaxMin, etc., which are usually stated by expected time to compute (ETC) matrix, machine availability time (MAT) vector, and completion time (CT) vector. One row of the ETC matrix contains the estimated execution times for a given task on each machine. Similarly, one column of the ETC matrix consists of the estimated execution times of a given machine for each task. Thus, for an arbitrary task $t_i$ and an arbitrary machine $m_j$, $ETC(i,j)$ is the estimated execution time of $t_i$ on $m_j$. Machine availability time $MAT(j)$ is the earliest time in which machine $m_j$ can complete the execution of all the tasks that have previously been assigned to it. The completion time for a task $t_i$ on machine $m_j$, $CT(i,j)$, is the machine availability time plus the execution time of task $t_i$ on machine $m_j$, $CT(i,j)=MAT(i,j)+ETC(i,j)$. The maximum $CT(i,j)$ value is known as the makespan. At the beginning, most of algorithms are designed to minimize the makespan with little consideration of other factors. However, multiobjective GTS has attracted more and more attentions with the deep research of GTS.

OLB assigns an unexecuted task to a currently available machine randomly. MET allocates an unexecuted task to a machine that gives it the minimum execution time. Both of them are regardless of the machine's current workload. So they can induce a heavy unbalancing workload. MCT absorbs their respective advantages, dispatching an unexecuted task to a machine that currently will result in minimum completion time. MinMin first calculates the set of minimum completion time for each task on all machines, and then the task with overall minimum completion time is selected and assigned to the corresponding machine. As a modification of MinMin, MaxMin is very similar to MinMin; just the task with overall maximum completion time is selected and assigned to the corresponding machine after obtaining the set of minimum completion time for each task.

Among these five algorithms, MinMin is the most outstanding one. Although MCT usually outperforms the OLB and MET, it can induce a larger makespan compared with MinMin. If there are several long tasks in many short tasks, MaxMin will outperform MinMin with a balance system workload. It is worth to mention that MinMin has been adopted in some practice projects. Here we make a summary of several improved algorithms based on MinMin in the literature. Wu *et al*. [11] presented a segmented MinMin algorithm which can make a more balancing workload than MinMin and MaxMin do. Taking QoS requirements of GTS into consideration, He *et al*. [12] proposed a QoS-guided MinMin algorithm, where both tasks and resources were sorted by different QoS levels first, and then tasks were selected and assigned to those resources whose QoS levels were equal or greater than those of tasks. Compared with the original MinMin, their algorithm can improve resource utilization ratio, satisfy user's QoS requirements, and make a good synthetical performance. This is also the main objective of GTS we purchase all the time. In 2007, Etminani *et al*. [13] introduced a new MinMin MaxMin selective algorithm based on MinMin and MaxMin. It transferred between the two algorithms based on the standard deviation of the expected completion time of tasks on resources. Their experimental results showed that the new algorithm can lead to a significant performance gain for a variety of scenarios. And then in 2008, Venugopal and Buyya [14] proposed a two-phase GTS model for scheduling an application composed of a set of independent tasks. In the first part, they applied a SCP-based heuristic to match tasks to resources, and the second part was tackled by extending MinMin and suffrage to schedule the set of distributed data-intensive tasks.

### *3.1.2 Heuristic Intelligent Meta-task GTS Algorithms*

Heuristic intelligent algorithms are inspired by some principles of nature such as biological evolution, physical process, and human thinking. At present, such algorithms, which are used in GTS, mainly include genetic algorithm (GA), simulated annealing (SA), genetic simulated annealing (GSA), tabu (tabu Search), ant colony optimization (ACO), particle swarm optimization (PSO), etc. As a basic work, it is necessary to use a number of different presentations to encode solutions onto chromosomes or individuals in such algorithms. The most commonly used presentations are binary, numeric, and symbolic. Taking the numeric presentation as an example, an individual is a $m \times 1$ vector, where position $i(0 \leq i < m)$ represents task $t_i$, and the entry in position $i$ is the machine to which the task has been mapped. Next, these main algorithms are introduced respectively.

*(1) Genetic algorithm***:** GA is one typical branch of evolutionary algorithms inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. It operates on a population of potential solutions, applying the principle of survival of the fittest to produce successively exact or approximate solutions to the given problems. Its main procedures can be depicted as Figure 3.

The GA has been applied for task scheduling in HC environments since the late 1990s [15]. In [16], Braun *et al*. made a comparison of 11 static algorithms; their simulations showed that the GA always gave the best results than other algorithms, and MinMin was the second best. Many efforts on GAs used in GTS have been done in the past years. In 2007, Carretero *et al*. [17] presented

an extensive study on the usefulness of GAs for designing efficient grid schedulers when both makespan and flow time were minimized. And then, Priya *et al*. [18] proposed a four-task-level fault tolerance technique including retry, alternate resource, check point, and replication. The GA for GTS with fault tolerance was presented using check point. In 2008, Yuan *et al*. [19] introduced an improved adaptive GA combined with neighborhood search, and their simulations showed that the proposed algorithm could greatly improve the performance of GTS.

*(2) Simulated annealing algorithm:* SA is a probabilistic heuristic for the optimization problems. It aims to merely find an acceptably good solution in a fixed amount of time rather than the best possible solution. Its inspiration comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heating causes the atoms to become unstuck from their initial positions and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one. Its basic procedure is shown in Figure 4.

SA has proved to be a flexible search method and can be successfully applied to the majority of practice problems. In 2006, Fidanova [20] introduced a task scheduling algorithm for GTS based on SA, and better results were obtained compared with ACO. And Kazem *et al*. [21] proposed a modified SA for scheduling independent tasks in grid environment in 2008. Their experimental results showed that it can improve the performance of static instances compared to the results of other algorithms reported in the literature.

*(3) Tabu search algorithm***:** Tabu uses a local or neighbourhood search procedure to iteratively move from a solution $x$ to another solution $x'$ in the neighborhood of $x$ until some stopping criterions have been satisfied. The basic procedures of tabu are illustrated in Figure 5.

To explore regions of the search space that would be left unexplored by the local search procedure, tabu modifies the neighborhood structure of each solution as the search progresses. The new neighborhoods are determined through the use of memory structures. The most important type of memory structure used to determine the solutions admitted to the neighborhood of $x$ is the tabu list. In its simplest form, a tabu list is a short-term memory which contains the solutions that have been visited recently. Its applications on GTS can be found in [16,22].

*(4) Ant colony optimization algorithm:* ACO is a new heu-

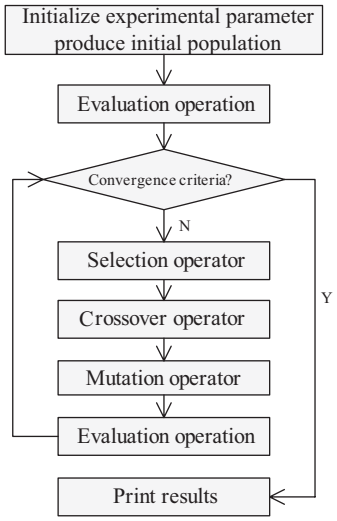



**Figure 3:** The basic process of GA on GTS.

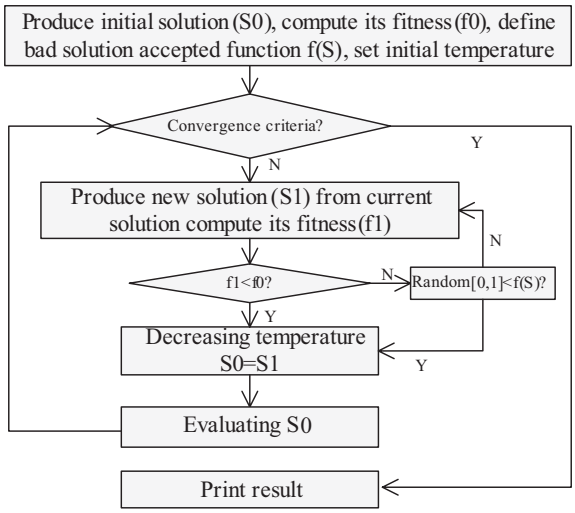



**Figure 4:** The basic process of the SA algorithm on GTS.

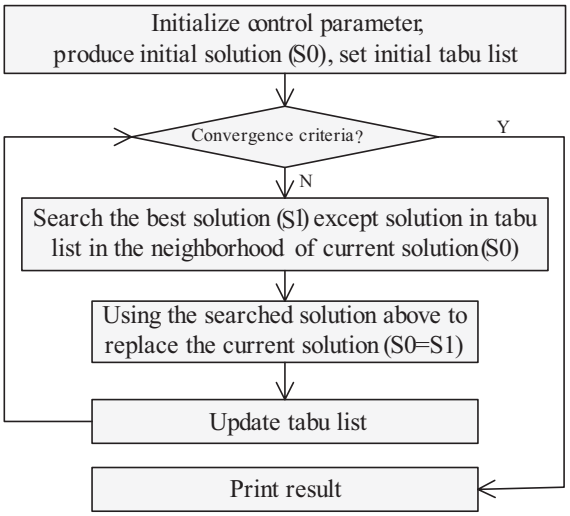



**Figure 5:** The basic process of the TS algorithm on GTS.

ristic algorithm. It is based on the behavior of real ants. When the blind insects, such as ants, look for food, every moving ant lays some pheromone on the path; then the pheromone on a shorter path will be increased quickly, and the quantity of pheromone on every path will affect the possibility of other ants to select the path. At last all the ants will choose the shortest path. ACO has been successfully used to solve many NP problems, such as TSP, job-shop scheduling, and graph coloring.

In 2003, Xu *et al*. [23] designed an ACO algorithm for GTS. When a resource registers itself into the grid, it is asked to submit its performance parameters such as number of PE, MIPS of each PE, etc. And then the resource monitor tests these parameters for validation and initializes the pheromone on each link by Equation (1), where $m$ is the number of PE, $p$ is the MIPS of one PE, $c$ is the size of parameters, and $s_j$ is parameter transfer time from $j$ to the resource monitor. While a new resource joins the grid, or a resource breaks down, or a task is assigned, or there are tasks returned, the pheromone on the corresponding path will be updated by Equation (2), where $\Delta\tau_j$ is the increment or decrement of pheromone on the path from schedule center to resource $j$, $\rho$ is the permanence of pheromone ($0 \le \rho \le 1$), and $1-\rho$ is the evaporation of the pheromone. The probability of task assignment to every resource will be recomputed by Equation (3), where $\tau_j(t)$ is the pheromone intensity on the path from the schedule center to resource $j$, $\eta_j$ is the innate performance of the resource, that is, $\tau_j(0)$, $\alpha$ is the importance of pheromone, and $\beta$ is the importance of resource innate attributes:

$$\tau_j(0) = m.p + c/s_j \tag{1}$$

$$\tau_j^{new} = \rho.\tau_j^{old} + \Delta\tau_j \tag{2}$$

$$p_j^k(t) = \begin{cases} \dfrac{\left[\tau_j(t)\right]^{\alpha}\left[\eta_j\right]^{\beta}}{\sum_u \left[\tau_u(t)\right]^{\alpha}\left[\eta_j\right]^{\beta}}, & j,u \in R \\ 0, & others \end{cases} \tag{3}$$

Ritchie *et al*. [24] presented a hybrid ACO for scheduling independent jobs in HC environments, which can consistently find better schedules for several benchmark problems than other techniques. In 2007, Lorpunmanee *et al*. [25] described an ACO algorithm for dynamic job scheduling in grid environment. Compared with FCFS, EDD, and ERD, ACO can efficiently and effectively allocate jobs to proper resources. And then, Liu *et al*. [26] proposed an improved ACO called adaptive ACO for GTS, which was more efficient than the original algorithm both in task scheduling efficiency and resource load.

*(5) Particle swarm optimization algorithm:* PSO is a population-based stochastic optimization technique inspired by social behavior of birds. It contains a swarm of particles in which each particle includes a potential solution. In contrast to evolutionary computation paradigms such as GA, a swarm is similar to a population, while a particle is similar to an individual. The particles fly through a multidimensional search space in which the position of each particle is adjusted according to its own experience and the experience of its neighbors. The PSO system combines local search methods (through self-experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation [27]. Generally, the velocity and position of each particle can be updated by Equations (4) and (5) [28,29]:

$$v_{id}(t+1) = \omega v_{id}(t) + c_1 r_1\left[p_{id} - x_{id}(t)\right] + c_2 r_2\left[p_{gd} - x_{id}(t)\right] \tag{4}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{5}$$

where $\omega$ is the inertia factor; $r_1$ and $r_2$ are random numbers between 0 and 1; $c_1$ and $c_2$ are the acceleration constants; $p_{id}$ is the best location of particle $i$ in a $d$-dimensional space; and $p_{gd}$ is the best position of the entire group.

In 2008, Bu *et al*. [30] described an improved PSO algorithm with a discrete coding rule for GTS. The experimental results showed that the improved PSO was stable and presented low variability. And it outperformed MaxMin by makespan and other performance. Later Izakian *et al*. [31] introduced a PSO algorithm for scheduling meta-task in distributed HC systems to minimize makespan and the proposed algorithm obtained higher performance than other compared techniques did.

Recently, some new heuristic intelligent algorithms are also introduced to solve GTS problem, such as AFSW [32] and HNN [33]. Besides, hybrid algorithms are also one of hotspots of GTS. As we know, GSA is a typical algorithm for the hybrid GTS algorithm. In 2006, Zheng *et al*. [34] proposed a parallel GSA algorithm combined with the advantages of GA and SA for GTS. The presented algorithm was superior to the genetic algorithm and simulated annealing according to the analysis and experiment result.

### 3.2 DAG GTS Algorithms

At first, GTS mainly focused on meta-task applications. So many proposed algorithms do not consider the priority constraint relation and data dependence between tasks. However, with the development of grid technology, its application ranges become more and more wide. The involved applications are also becoming more complex and diverse. So GTS inevitably needs to pro-

cess some interdependent tasks. It is worth to mention that it is the original intention that grids are treated as a generalized distributed parallel computing platform.

As mentioned above, many complex applications, which consist of interdependent tasks that cooperate in order to solve a particular problem, can be presented in the form of DAG. In recent years, many efforts have progressed in the DAG GTS algorithm. Especially, grid workflow scheduling opens a new road for solving the problem. In general, a workflow can be represented as a DAG or a non-DAG. In the DAG workflow, the structure can be classified as sequence, parallelism, and choice. In addition to all patterns contained in a DAG workflow, a non-DAG workflow also includes the iteration structure in which sections of workflow tasks in an iteration block are allowed to be repeated [35]. To date, many grid workflow scheduling methods are based on the DAG workflow.

Yu *et al.* [36] investigated several existing workflow scheduling algorithms deployed in the different grid environments. They classified those algorithms into two major types, namely, best-effort-based and QoS-constraint-based scheduling. The best-effort-based scheduling attempts to minimize the execution time ignoring other factors such as the monetary cost of accessing resources and various users' QoS satisfaction levels. On the other hand, QoS-constraint-based scheduling attempts to minimize performance under the most important QoS constraints, for example, time minimization under budget constraints or cost minimization under deadline constraints.

Here we survey some typical DAG GTS algorithms in terms of the same classifications used in meta-task GTS algorithms, including traditional deterministic algorithms and heuristic intelligent algorithms.

### *3.2.1 Traditional Deterministic DAG GTS Algorithms*

Topcuoglu *et al*. [37] proposed a well-established list scheduling algorithm called heterogeneous earliest finish time (HEFT), which assigned higher priority to the workflow task having a higher rank value. The rank value is calculated based on the average execution time for each task and average communication time between resources of two successive tasks, where the tasks in the critical path get comparatively higher rank values. Later in 2004, Sakellariou and Zhao [38] proposed a hybrid heuristic for scheduling DAG on heterogeneous systems, and the experimental results demonstrated its good performance behaviors. In 2006, Du *et al*. [4] addressed a fuzzy clustering-based scheduling heuristic (FCBSH), which took both the heterogeneity of resources and priority constraint relation of tasks into account simultaneously. In 2008, considering time and cost parameter, Fard *et al.* [39] proposed a new list heuristic algorithm named TCI (time and cost improvement) for workflow applications modelled as DAG; TCI is an extended version for PETS, which is DAG-based algorithm for the optimization of makespan. Their experimental results showed that TCI can meet low time as well as low cost. Besides, some algorithms, initially designed for scheduling parallel independent tasks, are also applied to allocate tasks onto resources in certain workflow applications, such as MinMin, MaxMin, and suffrage [40,41]. These algorithms usually group workflow tasks into several independent tasks and consider tasks only in the current group.

### *3.2.2 Heuristic Intelligent DAG GTS Algorithms*

Based on GA, Yu *et al.* [42] proposed a budget constraint workflow scheduling approach modelled as DAG, where a 2D string is used to encode a schedule as well as new genetic operators are designed. Their algorithm can minimize the execution time while meeting a specified budget. In 2008, Aziz and El-Rewini [43] presented a matching and scheduling algorithm based on SA named as PRISM-SA, which uses SA and concepts from tabu search to decrease the processing time. PRISM-SA made a drastic improvement in job processing time than the original PRISM and HEFT. ACO and PSO have also been applied to map tasks to resources in the applications represented by DAG in the recent researches. Chen *et al.* [44] designed an ACO algorithm to tackle the workflow scheduling problem concerning about the users' QoS requirements as well as minimizing the cost. Two kinds of pheromones and three kinds of heuristic information are defined to guide the search direction of the ants for the bicriteria problem and the information of partial solutions are applied to modify the bias of ants to avoid inferior choices. And this algorithm performs better than the deadline-MDP algorithm in the experiments. Two years later in 2009, Chen *et al.* [45] applied ACO to solve another grid workflow application, namely, the time-varying workflow, in which the topologies of DGA change over time. A nine-task gird workflow with four topologies is used to test the performance of and the experimental results demonstrate the effectiveness and robustness of the algorithm. Tao *et al.* [46] put forward a novel PSO algorithm called rotary hybrid discrete particle swarm optimization (RHDPSO) to solve the multi-dimensional QoS constrained grid workflow scheduling problem described by DAG, in which double extremums are disturbed by the method of random time sequence based on rotation discretization, to overcome premature convergence and local optimum. The simulation results show that the RHDPSO algorithm has fast convergence, high precision, and strong robustness, and can effectively restrain premature convergence compared with DPSO. Moreover, Li *et al.* [47] proposed a scheduling algorithm of multiobjective optimal grid workflow scheduling with

QoS constraints based on the multiobjective particle swarm optimization (MOPSO) algorithm, which outperforms a grid workflow scheduling algorithm based on the NSGA-II algorithm.

Compared with mate-task GTS algorithms, there are many new challenges which need to be addressed for DAG GTS algorithms, such as large data transmission across various data communication links and order restrictions of task execution. Besides, these algorithms pay more attentions to user's QoS requirements such as time, cost, fidelity, reliability, security, and so on. As we have seen, many algorithms have been proposed in the literature. However, for the dynamic nature of grid environments and the complexity of scheduling interdependent tasks, it is difficult to find an approach to satisfy all kinds of requirements, and there are still many problems that need to be resolved in this aspect.

### 3.3 Comparative Analysis of Different Algorithms

The comparative analysis of time complexity and characteristics among the algorithms mentioned above on the basis of some representative algorithms is depicted in Table 1. In general, meta-task GTS algorithms are simpler than DAG GTS algorithms for their just considering the allocations of independent tasks without priority constraints. Traditional deterministic algorithms, applied in both meta-task and DAG GTS algorithms, can generate a feasible solution in a polynomial time, whereas the scheduling time needed to produce a good quality solution required by heuristic intelligent algorithms is significantly higher because they usually exploit the feasible solution space in a number of iterations according to different guided random search techniques. However, intelligent algorithms can produce better quality solutions and be suitable for different application scenarios. Unlike this point, the traditional deterministic algorithms are usually designed for a particular application. Sometimes these two types of scheduling approaches are incorporated to generate a satisfactory solution in shorter time.

## 4. Research Directions of GTS

GTS is the key issue of grid computing. Up to now, all kinds of technologies and solutions on GTS have been proposed over the past years. According to our long-term tracking and analyzing, we believe the following several research directions will play an important role in the future:

*(1) Constrained multiobjective GTS:* As mentioned above, many efforts have progressed both in meta-task GTS algorithms and DAG GTS algorithms in the past few years. Besides, the optimization objectives of GTS have also been changing. Initially, makespan was the main consideration, and then deadline and budget were taken into account based on the market/economic scheduling model. Up to now, all of such objectives have been formulated as the requirements of QoS such as time, cost, fidelity, reliability, efficiency, and security, either from the perspective of the system or from the users. At present, these different objectives usually are aggregated into a single objective in the form of utility function, which are introduced in Ref [48]. However, this requires some prior knowledge to determine initial parameters, and it is difficult to obtain some information for the nature of dynamic of grid computing environment. Besides, considering the complexity of grid workflow scheduling, some new constrained multiobjective GTS methods are needed to make sufficient optimization among the entire objectives in the future work.

*(2) Mobile grid computing GTS:* In recent years, with the proliferation of wireless mobile devices and the development of wireless network technology, mobile grids/ad hoc grids/pervasive grids have been emerging as a

**Table 1: Comparison analysis of mentioned grid task scheduling algorithms**

| Categories | Representative algorithm | Time complexity | Characteristics analysis |
|---|---|---|---|
| Meta-task GTS algorithms | | | |
| Traditional deterministic algorithms | MinMin | $O(m^2n)$ | Scheduling decision based on independent tasks, simple implementation, and the scheduling time of obtaining a feasible solution is lower |
| Heuristic intelligent algorithms | GA | Higher | Scheduling decision based on independent tasks, global solution obtained by combining current best solutions and exploiting new search space iteratively, high robustness but longer time needed for higher quality solutions |
| DAG GTS algorithms | | | |
| Traditional deterministic algorithms | HEFT | $O(k^2n)$ | Scheduling decision based on interdependent tasks, the strategies required to set the priority of tasks, lower time complexity but restriction of the scale of applications |
| Heuristic intelligent algorithms | ACO | Higher | Scheduling decision based on interdependent tasks, more suitable for the multiobjective and multiconstraint DAG-based scheduling, and higher quality scheduling can be achieved but longer time is also required |

$m$ is the number of tasks, $k$ the number of nodes, and $n$ the number of $PU_S$.

new computing paradigm, which is the conjunction of grid computing and pervasive/mobile computing. Traditional grid infrastructures are mostly based on wired network resources owned by various individuals and/or institutions, structured in virtual organizations, which are subjected to specific sharing policies. Unlike the classical grid system, mobile grids enable wireless and mobile users to share computing resources, services, and information and include wirelessly networked portable devices (laptops, PDAs, mobile phones, wireless sensors, etc.). The research of mobile grid computing can make grid system more user-friendly and pervasive. Some issues and architectures have been proposed in this field [49-54]. However, the adaptation of grid technology to ad hoc networks is not straightforward, and there still exist numerous difficulties needing to be conquered, such as mobile resource discovery, power consumption, QoS, security, etc. Also, task scheduling is vital for the mobile grids and more attention needs to be paid to task replication/migration for the instability of mobile devices. This direct is attracting more and more attention.

*(3) Agent-based GTS:* Agent architectures offer valuable techniques to provide the autonomy and flexibility required in highly dynamic and heterogeneous environments. Multiagent systems imply coordination and cooperation among their agents. The agent-based grid is intended to provide a completely distributed environment within which agent systems and individual agents can participate in a broader community of agents, utilizing services and capabilities provided by the other participants or the grid itself. The combination of intelligent agents and multiagent approaches can be applied to both local grid resource scheduling and global grid load balancing. Although much work [55-59] has been done in this aspect, but there still exist many challenges. Especially, it is very promising to construct and deploy the self-government distributed grid system using the technology of agent.

As you have seen, the issue of jointly using several technologies to solve certain problems has recently received more and more attention. In addition to the points mentioned above, many other similar studies have also been initiated such as grid-based DDM (distributed data mining) [60-62] and the usage of game theory [63-65]. So the hybrid approaches will play an important role in the future work.

## 5. Conclusion

In this paper, we have done a detailed analysis and discussion on the current main GTS algorithms, and cleared several researching branches of GTS and their respective characteristics. In addition, several future researching directions on GTS are summed up and analyzed finally. They maybe are useful for fresh researchers as references or overviews of the current GTS field.

## 6. Acknowledgment

This research was partly supported by Natural Science Foundation from Nanjing University of Information and Science Technology (20080302), Natural Science fund for colleges and universities in Jiangsu Province (08KJD520018), Jiangsu overseas study scholarship, and Jiangsu Youth Project.

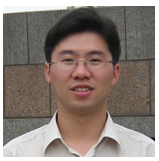
## References

1. H. Luo, D. Mu, Z. Deng, and X. Wang. "A review of job scheduling for grid computing," Application Research of Computer, vol. 22, no. 5, pp.16-9, May. 2005.
2. J. D. Ullman. "NP-complete scheduling problems," Journal of Computer and System Sciences, vol. 10, issue 3, pp. 384-93, Jun. 1975.
3. J. Li. "Research on tasks DAG scheduling algorithm based on grid," PhD thesis, Central South University, Hunan, China, 2008.
4. X. Du, C. Jiang, G. Xu, and Z. Ding. "A grid DAG scheduling algorithm based on fuzzy clustering," Journal of Software, vol. 17, no. 11, pp. 2277-88, Nov. 2006.
5. M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," Journal of Parallel and Distributed Computing, vol. 59, Issue 2, pp. 107-31, Nov. 1999 .
6. G. Ritchie and J. Levine. "A fast, effective local search for scheduling independent jobs in heterogeneous computing environments," in the 22nd Workshop of the UK Planning and Scheduling Special Interest Group, Glasgow, UK, pp. 178-83, 2003.
7. N. Fujimoto and K. Hagihara. "A comparison among grid scheduling algorithms for independent coarse-grained tasks," in SAINT 2004 Workshop on High Performance Grid Computing and Networking, Tokyo, Japan, 2004, pp. 674-80.
8. H. D. Kim and J. S. Kim. "An online scheduing algorithm for grid computing systems," in the 2th International Workshop on Grid and cooperative computing, GCC 2003, Shanghai, China, pp. 34-9, Dec. 2004.
9. P. Luo, K. Lü and Z. Z. Shi. "A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems," Journal of Parallel and Distributed Computing, vol. 67, issue 6, pp. 695-714, Jun. 2007.
10. L. Y. Tseng, Y. H. Chin, and S. C. Wang. "The anatomy study of high performance task scheduling algorithm for Grid computing system," Computer Standards and Interfaces, vol. 31, issue 4, pp. 713-22, Jun. 2009.
11. M. Y. Wu, W. Shu, and H. Zhang. "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems," in 9th IEEE Heterogeneous Computing Workshop (HCW 2000), Cancun, Mexico, pp. 192-6, 2000.
12. X. S. He, X. H. Sun, and G. V. Laszewski. "QoS guided min-min heuristic for grid task scheduling," Journal of Computer Science and Technology, vol. 18, no. 4, pp. 442-51, 2003.
13. K. Etminani and M. Naghibzadeh. "A Min-Min Max-Min Selective Algorithm for Grid Task Scheduling," in 3rd IEEE/IFIP International Conference in Central Asia on Internet (ICI'07), Tashkent, Uzbekistan, pp. 1-7, 2007.
14. S. Venugopal and R. Buyya. "An SCP-based heuristic approach for scheduling distributed data-intensive applications on global grids," Journal of Parallel and Distributed Computing, vol. 68, issue 4, pp. 471-87, Apr. 2008.
15. L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski. "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach," Journal of Parallel and Distributed Computing, vol. 47, issue 1, pp. 8-22, Nov. 1997.

16. T. D. Braun, H. J. Siegelb, N. Beckc, and L. L. Bölönid, *et al*. "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," Journal of Parallel and Distributed Computing, vol. 61, issue 6, pp. 810-37, Jun. 2001.
17. J. Carretero, F. Xhafa, and A. Abraham. "Genetic algorithm based schedulers for grid computing systems," International Journal of Innovative Computing, Information and Control, vol. 3, no.5, pp.1053-71, Oct. 2007.
18. S. B. Priya, M. Prakash, and K. K. Dhawan. "Fault tolerance-genetic algorithm for grid task scheduling using check point," in GCC2007, Urumchi, China, pp. 676-80, 2007.
19. J. B. Yuan, J. M. Luo, and Z.Y. Su. "Strategy for tasks scheduling in grid combined neighborhood search with improved adaptive genetic algorithm based on local convergence criterion," in 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, pp. 9-13, 2008.
20. S. Fidanova. "Simulated annealing for grid scheduling problem," in IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06), Sofia, Bulgaria, pp. 41-5, 2006.
21. A. A. P. Kazem, A. Kazem, A. M. Rahmani, and H. H. Aghdam. "A Modified Simulated Annealing Algorithm for Static Task Scheduling in Grid Computing," in International Conference on Computer Science and Information Technology, Singapore, pp. 623-7, 2008.
22. R. Subrata, A. Y. Zomaya, and B. Landfeldt. "Artificial life techniques for load balancing in computational grids," Journal of Computer and System Sciences, vol. 73, issue 8, pp. 1176-90, Dec 2007.
23. Z. Xu, X. Hou, and J. Sun. "Ant algorithm-based task scheduling in grid computing," in Canadian Conference on Electrical and Computer Engineering (CCECE 2003), Montreal, Canada, vol. 2, pp. 1107-10, 2003.
24. G. Ritchie and J. Levine. "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments," in the 23rd Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2004), pp. 1-7, Dec. 2004.
25. S. Lorpunmanee, M. N. Sap, A. H. Abdullah, and C. Chompoo-inwai. "An ant colony optimization for dynamic job scheduling in grid environment," International Journal of Computer and Information Science and Engineering, vol. 1, pp. 207-14, 2007.
26. A. H. Liu and Z. Y. Wang. "Grid task scheduling based on adaptive ant colony algorithm," in International Conference on Management of e-Commerce and e-Government (ICMECG'08), Nanchang, China, pp. 415-8, 2008.
27. A. Salman, I. Ahmad, and S. Al-Madani. "Particle swarm optimization for task assignment problem," Microprocessors and Microsystems, vol. 26, issue 8, pp. 363-71, Nov. 2002.
28. Y. Shi and R. C. Eberhart. "A modified particle swarm optimizer," in IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, pp. 69-73, May. 1998.
29. Y. Shi and R. C. Eberhart. "Fuzzy adaptive particle swarm optimization," in the 2001 Congress on Evolutionary Computation (CEC2001), Seoul, Korea, vol. 1, 2001, pp.101-6.
30. Y. P. Bu, W. Zhou, and J. S. Yu. "An improved PSO algorithm and its application to grid scheduling problem," in 2008 International Symposium on Computer Science and Computational Technology, Shanghai, China, pp. 352-5, 2008.
31. H. Izakian, A. Abraham, and V. Snášel. "Metaheuristic based scheduling meta-tasks in distributed heterogeneous computing system," Sensors, vol. 9, pp. 5339-50, 2009.
32. S. Farzi. "Efficient job scheduling in grid computing with modified artificial fish swarm algorithm," International Journal of Computer Theory and Engineering, vol. 1, pp. 13-8, 2009.
33. C. F. Wang, H. Y. Wang, and F. C. Sun. "Hopfield neural network approach for task scheduling in a grid environment," in 2008 International Conference on Computer Science and Software Engineering, Wuhan, China, vol. 4, pp. 811-4, 2008.
34. S. J. Zheng, W. N. Shu, and L. Gao. "Task scheduling using parallel genetic simulated annealing algorithm," in 2006 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI 2006), Shanghai, China, vol. 1,pp. 46-50, 2006.
35. J. Yu and R. Buyya. "A taxonomy of workflow management systems for grid computing," Journal of Grid Computing, vol. 3, pp. 171-200, 2006.
36. J. Yu, R. Buyya, and K. Ramamohanarao. " Workflow scheduling algorithms for grid computing," Studies in Computational Intelligence, vol. 146, pp. 173-214, 2008.
37. H. Topcuoglu, S. Hariri, and M. Y. Wu. "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Transactions on Parallel and Distributed Systems, vol.13, issue 3, pp.260-74, 2002.
38. R. Sakellariou and H. Zhao. "A hybrid heuristic for DAG scheduling on heterogeneous systems," in the 13th Heterogeneous Computing Workshop (HCW 2004), Santa Fe, New, Mexico, USA, April 26, 2004.
39. H. M. Fard and H. Deldari. "An economic approach for scheduling dependent tasks in grid computing," in Proceedings of the 11th IEEE International Conference on Computational Science and Engineering, CSE Workshops, pp.71-6, 2008.
40. F. Berman, H. Casanova, A. Chien, K. Cooper, H. Dail, A. Dasgupta *et al*. "New Grid Scheduling and Rescheduling Methods in the GrADS Project," International Journal of Parallel Programming, vol. 33, pp. 209-29, 2005.
41. J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal *et al*., "Task scheduling strategies for workflow-based applications in grids," in IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005), vol. 2, pp. 759-67, 2005.
42. J. Yu and R. Buyya. "A budget constrained scheduling of workflow applications on utility grids using genetic algorithms," in 2006 Workshop on Workflows in Support of Large-Scale Science, Paris, France, Jun. 2006.
43. A. Aziz and H. El-Rewini. "On the use of meta-heuristics to increase the efficiency of online grid workflow scheduling algorithms," Cluster Computing, vol. 11, no. 4, pp. 373-90, Dec. 2008.
44. W. N. Chen, J. Zhang, and Y. Yu. "Workflow scheduling in grids: an ant colony optimizaton approach," in IEEE Congress on Evolutionary Computation (CEC2007), Singapore, Sep. 2007.
45. W. N. Chen, Y. Shi, and J. Zhang. "An ant colony optimization algorithm for the time-varying workflow scheduling problem in grids," in 2009 IEEE Congress on Evolutionary Computation, CEC 2009, Guangzhou, China, pp. 875-80, 2009.
46. Q. Tao, H. Y. Chang, Y. Yi, C. Q. Gu, and Y. Yu. "QoS constrained grid workflow scheduling optimization based on a novel PSO algorithm," in 8th International Conference on Grid and Cooperative Computing, GCC 2009, Guangzhou, China, pp. 153-9, 2009.
47. J. Z. Li, J. T. Zeng, J. W. Xia, M. H. Li, and C. X. Liu, "Research on grid workflow scheduling based on mopso algorithm", in Proceedings of the 2009 WRI Global Congress on Intelligent Systems, GCIS 2009, Jiangxi, China, vol. 1, pp. 199-203, 2009.
48. C. L. Li, and L. Y. Li. "A distributed multiple dimensional QoS constrained resource scheduling optimization policy in computational grid," Journal of Computer and System Sciences, vol.72, issue 4, pp.706-26, Jun. 2006.
49. M. Gaynor, M. Welsh, S. Moulton, A. Rowan, E. LaCombe, and J. Wynne. "Integrating wireless sensor networks with the grid," IEEE Internet Computing (special issue on the wireless grid), pp. 32-9, Jul–Aug. 2004.
50. A. Litke, D. Skoutas, K. Tserpes, and T. Varvarigou. "Efficient task replication and management for adaptive fault tolerance in mobile grid environments," Future Generation Computer Systems, vol. 23, issue 2, pp. 163-78, Feb. 2007.
51. J. K. Kim, H. J. Siegel, A. A. Maciejewski, and R. Eigenmann. "Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling," IEEE Transactions on Parallel and Distributed Systems, vol.19, issue 11, pp. 1445-57, 2008.

52. A. Coronato and G. D. Pietro. "MiPeG: A middleware infrastructure for pervasive grids," Future Generation Computer Systems, vol. 24, issue 1, pp. 17-29, Jan. 2008.
53. M. Vozmediano. "A hybrid mechanism for resource/service discovery in ad-hoc grids," Future Generation Computer Systems, vol. 25, issue 7, pp. 717-27, Jul. 2009.
54. M. Nekovee and R. S. Saksena. "Simulations of large-scale WiFi-based wireless networks: Interdisciplinary challenges and applications," Future Generation Computer Systems, vol. 26, issue 3, pp. 514-20, Mar. 2010.
55. C. L. Li and L. Y. Li. "Agent framework to support the computational grid," Journal of Systems and Software, vol. 7, pp.177-87, 2004.
56. Y. Gil. "On agents and grids: Creating the fabric for a new generation of distributed intelligent systems," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 4, issue 2, pp. 116-23, Jun. 2006.
57. D. Chen, G. K. Theodoropoulos, S. J. Turner, W. Cai, R. Minson, and Y. Zhang. "Large scale agent-based simulation on the grid," Future Generation Computer Systems, vol. 24, issue 7, pp. 658-71, Jul. 2008.
58. L. Han and D. Berry. "Semantic-supported and agent-based decentralized grid resource discovery," Future Generation Computer Systems, vol. 24, issue 8, pp. 806-12, Oct. 2008.
59. M. S. Pérez, A. Sánchez, J. H. Abawajy, V. Robles, and J. M. Peña. "An agent architecture for managing data resources in a grid environment," Future Generation Computer Systems, vol. 25, issue 7, pp. 747-55, Jul. 2009.
60. P. Luo, K. Lü, Z. Shi, and Q. He. "Distributed data mining in grid computing environments", Future Generation Computer Systems, vol. 23, issue 1, pp. 84-91, Jan. 2007.
61. V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, J. Kindermann *et al*. "Grid-enabling data mining applications with DataMiningGrid: An architectural perspective," Future Generation Computer Systems, vol.24, issue 4, pp. 259-79, Apr. 2008.
62. M. Swain, C. G. Silva, N. Loureiro-Ferreira, V. Ostropytskyy, J. Brito, O. Riche *et al*. "P-found: Grid-enabling distributed repositories of protein folding and unfolding simulations for data mining," Future Generation Computer Systems, vol.26, issue.3, pp. 424-33, Mar. 2010.
63. Y. K. Kwok, K. Hwang, and S. Song. "Selfish grids: game-theoretic modeling and NAS/PSA benchmark evaluation," IEEE Transactions on Parallel and Distributed Systems, vol. 18, issue 5, pp. 621-36, 2007.
64. R. Subrata, A.Y. Zomaya, and B. Landfeldt. "Game-theoretic approach for load balancing in computational grids," IEEE Transactions on Parallel and Distributed Systems, vol.19, issue 1, pp. 66-76, 2008.
65. S. U. Khan and I. Ahmad. "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," IEEE Transactions on Parallel and Distributed Systems, vol. 20, issue 3, pp. 346-60, 2009.


## AUTHORS



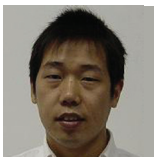

**Tinghuai Ma** is an associate professor in Computer Sciences at Nanjing University of Information Science & Technology, China. He received his Bachelor (HUST, China, 1997), Master (HUST, China, 2000), PhD (Chinese Academy of Science, 2003) and was Post-doctoral associate (AJOU University, 2004). From Nov. 2007 to Jul. 2008, he visited Chinese Meteorology Administration. From Feb.2009 to Aug. 2009, he was a visiting professor in Ubiquitous computing Lab, Kyung Hee University. His research interests are in the areas of Data Mining and Privacy Protected in Ubiquitous System, Grid Computing. His research interests are data mining, grid computing, ubiquitous computing, privacy preserving etc. He has published more than 50 journal/conference papers. He is a member of IEEE.

**E-mail:** thma@nuist.edu.cn



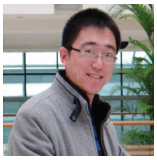
**Qiaoqiao Yan** received her Bachelor degree in Computer Science and Engineering from Nanjing University of Information Science & Technology, China in 2009. Currently, she is a candidate for the degree of Master of Computer Science and Engineering in Nanjing University of Information Science & Technology. Her research interests include Grid Computing, Quantum Computing etc.

**E-mail:** xinfeiyanrjgc@126.com


**Wenjie Liu** is an assistant professor of Computer Sciences at Nanjing University of Information Science & Technology, China. He received his Bachelor (WuHan University, China, 2001), Master (WuHan University, China, 2004). Now, he is a doctoral candidate of southeast University, China. His research interests are in the areas of Grid computing and Quantum computing.

**E-mail:** wenjieliu@nuist.edu.cn


**Donghai Guan** received his B.S. in College of Automation from Harbin Engineering University (HEU), Harbin, China in 2002. He got his M.S. degree in Computer Science from Kumoh National Institute of Technology (KIT), Gumi, South Korea in 2004. He got his Ph.D. degree in Computer Science from Kyung Hee University, South Korea in 2009. From 2009, he was a Post Doctoral Fellow at Computer Science Department, Kyung Hee University. His research interests are Machine Learning, Pattern Recognition, Data Mining, Activity Recognition, and Trust modeling.

**E-mail:** donghai@oslab.khu.ac.kr



**Sungyoung Lee** received his B.S. from Korea University, Seoul, Korea. He got his M.S. and Ph.D. degrees in Computer Science from Illinois Institute of Technology (IIT), Chicago, Illinois, USA in 1987 and 1991 respectively. He has been a professor in the Department of Computer Engineering, Kyung Hee University, Korea since 1993. He is a founding director of the Ubiquitous Computing Laboratory, and has been a.liated with a director of Neo Medical ubiquitous-Life Care Information Technology Research Center, Kyung Hee University since 2006. Before joining Kyung Hee University, he was an assistant professor in the Department of Computer Science, Governors State University, Illinois, USA from 1992 to 1993. His current research focuses on Ubiquitous Computing and applications, Context-aware Middleware, Sensor Operating Systems, Real-Time.

**E-mail:** sylee@oslab.khu.ac.kr


DOI: 10.4103/0256-4602.76138; Paper No TR 448_09; Copyright © 2011 by the IETE