Future Generation Computer Systems 28 (2012) 513–524

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A probabilistic task scheduling method for grid environments

Reza Entezari-Maleki*, Ali Movaghar

Performance and Dependability Laboratory, Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 19 February 2011
Received in revised form
8 September 2011
Accepted 17 September 2011
Available online 28 September 2011

Keywords:
Probabilistic scheduling
Grid environment
Mean response time
Discrete time Markov chain

ABSTRACT

This paper presents a probabilistic task scheduling method to minimize the overall mean response time of the tasks submitted to the grid computing environments. Minimum mean response time of a given task can be obtained by finding a subset of appropriate computational resources to service the task. To achieve this, a discrete time Markov chain (DTMC) representing the task scheduling process within the grid environment is constructed. The connection probabilities between the nodes representing the grid managers and resources can be considered as transition probabilities of the obtained DTMC. Knowing the mean response times of the managers and resources, and finding fundamental matrix of the DTMC, the mean response time related to each of the absorbing DTMCs existing inside the overall DTMC can be computed. Minimizing the obtained mean response times and taking into account the probability constraints in each of the absorbing DTMCs, a nonlinear programming (NLP) problem is defined. Solving the NLP problem, the connection probabilities between the managers and resources are obtained. Finally, using the connection probabilities, the best scheduling path within the environment and the minimum mean response time of a particular task can be achieved. In a case in which there is only one optimal scheduling choice within the environment, the proposed method can deterministically find such scheduling by assigning zero or one to the connection probabilities. Results obtained from evaluating the proposed method on the hypothesis and real grid environments show the preference of the proposed method compared to the other methods in minimizing both the overall mean response time of the tasks and total makespan of the environment.

1. Introduction

Task scheduling is one of the well-known problems in distributed computing systems such as grid environments [1–3]. Since the grid resources are very heterogeneous and have different processing capabilities, the task scheduling problem becomes more important in grids. Many static and dynamic scheduling algorithms have been proposed to schedule grid tasks among the resources to achieve the required amount of quality of service (QoS) [3,4]. The most important QoS parameters which can be studied in grid environments are as follows: performance of the environment, availability of the resources, service reliability, security of the grid services, throughput of the grid systems, overall completion time of the tasks, tasks' execution times, tasks' waiting times and so forth [2,4–6].

As one of the most important QoS measures in grid environments, the response time of a task needs to be studied. Response time of a specific task is important, because of its ability to describe the performance of the system from the user's perspective. Since the success of most of the services provided by distributed systems depends on user satisfaction, such a user-oriented metric may be a more appropriate way to rate a system's effectiveness [7–9]. Therefore, minimizing the response time of a particular task can be considered as a goal of scheduling algorithm in grid environments. But, on the other hand, in grid environments, it may be more important that the system's behavior be consistent, rather than faster overall but occasionally inconsistent [7]. Hence, considering the mean response time of the tasks [8,10–12] in grid environments can be more interesting than considering the response time of the tasks. In addition to the mean response time of the tasks which is one the user-oriented performance metrics in grid environments, total makespan of the grid environment can be considered in the scheduling algorithms [4–6,12–15]. The total makespan of the grid environment is one of the system-oriented performance measures which is defined as a largest makespan of the grid resources existing in the environment. The makespan of a resource is also defined as the total completion time of the tasks assigned to that resource. Minimizing the total makespan of the grid environment, the throughput of the environment is increased, accordingly.

Because of resource heterogeneity and application diversity in grid environments, discovering available resources and selecting a suitable subset of those resources are very important to achieve high performance and high throughput computing. Searching

* Corresponding author. Tel.: +98 21 6616 6678.
E-mail addresses: entezari@ce.sharif.edu, reza.em@gmail.com (R. Entezari-Maleki), movaghar@sharif.edu (A. Movaghar).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.09.005

for suitable resources to service the submitted tasks is critical mission for scheduling algorithms in grid environments. Definition of the suitable resource is very depended on the goal of the scheduling. In this paper, the goal of the scheduling is to minimize the overall mean response time of the submitted tasks and total makespan of the environment. Therefore, grid manager should find an appropriate resource to service the tasks, considering the completion times of them. The grid manager in a grid environment is responsible for locating, reserving, allocating, monitoring, and de-allocating one or more computational resources for an application [4,10,13,16]. The proposed scheduling method can be used in each of the grid managers to find a resource with low mean response time for the tasks submitted to the grid environment. To achieve this, a state transition diagram representing the grid environment is constructed. Grid managers and resources are shown by vertices and communication links among the managers and resources are shown by edges in the state transition diagram. Each of the edges is labeled by a real number between zero and one to show the connection probability between vertices. If a probability of the edge between node *a* and node *b* becomes one, then node *a* can send data to node *b* and if this number becomes zero, there is no data transmission between these nodes. Based on the special conditions existing in the state transition diagram representing the grid environment which is further explained later, the state transition diagram can be considered as an absorbing discrete time Markov chain (DTMC) [17,18]. Applying quantitative analysis to the absorbing DTMC and considering the aim of the proposed method which is finding the connection probabilities to minimize the mean response times of the tasks, a nonlinear programming (NLP) problem is defined. Solving this problem, the path of the scheduling, probability matrix of the DTMC, target resource(s) for executing the tasks, the minimum mean response time of the tasks and minimum total makespan of the environment can be found.

The remaining part of the paper is organized as follows. Section 2 introduces the related research works done on the probabilistic scheduling in various systems. In Section 3, the basic model of the grid environment used in this paper is presented. Section 4 proposes the probabilistic task scheduling method and Section 5 provides two illustrative examples to show the application of the proposed method. In Section 6, performance evaluation is done to compare the proposed method against to the other similar scheduling approaches. Finally, Section 7 concludes the paper and presents future work.

2. Related work

A lot of probabilistic scheduling algorithms have been proposed to schedule jobs and requests among various servers. Considering the architecture and characteristics of the grid environments, these probabilistic algorithms cannot be applied to the grids properly. In the follow, we describe some of the probabilistic scheduling schemes proposed for different environments.

Bestavros and Spartiotis [19] have presented a probabilistic job scheduling heuristic in distributed systems to meet tasks' deadlines. The algorithm tries to schedule a submitted task locally to meet its deadline. If that is not possible, it tries to find another node with high probability of success. To do this, when a sporadic task arrives at a node and the node cannot guarantee the execution of the task, it starts looking for a suitable node. First, it selects probabilistically a node from the light-load category or the medium-load category. After a category has been selected, the nodes in this category are considered to be allocated to the task. In some situations, there is a probability to find more than one suitable node for executing the submitted task. In this case, the target node is selected probabilistically. The algorithm presented in [19] only tries to increase the number of the sporadic tasks which are accepted for execution. In other words, it does not consider response time of a particular task or completion time of a bag of tasks which are very interesting issues in grid computing environments.

Anand et al. [20] have presented a probabilistic load scheduling algorithm within distributed systems. The proposed algorithm in [20] considers local and global tasks for each of the systems existing in the network and tries to find the best distribution of the tasks among the resources. Since different priorities for each of the local and global tasks are considered, the model uses a priority queuing optimization model to formulate a NLP problem to find the scheduling probabilities. Applying the scheduling probabilities obtained from solving the NLP problem to the global tasks arrival rate, load distributed among the resources can be balanced. The algorithm proposed in [20] cannot be applied to the grid environments with distributed managers, because in this model, all of the global tasks are submitted to one of the managers and users are forced to deliver their own tasks to the central terminal.

Jiang et al. [21,22] have presented two probabilistic priority scheduling disciplines for high speed and multi-service networks. In [21], the starvation of low priority packets against to the high priority ones is considered and a solution to avoid this problem is presented. To achieve this, a parameter is assigned to each of the priority queues in each of the servers. This parameter determines the probability in which its corresponding queue is served when the queue is sampled by the server. Using this mechanism, a new packet scheduling discipline named probabilistic priority (PP) is presented. The PP improves the performance and average throughput of the scheduling by probabilistically assigning input tasks to the servers. Similar to the discipline proposed in [21], another PP scheduling for multi-service networks has been presented in [22]. Tham et al. [23] have extended the PP scheduling discipline to the multi class cases. In this version of PP scheduling discipline, various classes for each of the input queues are considered and relationship between the average queuing delay for each class and its probabilities are derived.

Salami and Chan [24], Salami et al. [25] have presented two probabilistic scheduling algorithms for IP traffic. In [24], an iterative probabilistic scheduling (IPS) algorithm has been presented to model virtual output queuing strategy in each of the input queues implemented in routers. IPS algorithm considers the first in first out (FIFO) mechanism for each of the output queuing strategies. First, IPS orders the packets using estimating the transmission bandwidth and the waiting time. Then the weight of each packet is calculated using these two parameters. The algorithm uses these weights to determine the probability of transmitting each of the packets retrieved during a specific time slot. After determining the related probabilities for each of the packets, the packets can be scheduled among output ports. In [25], the proposed IPS is extended to support other mechanisms such as PIM and iSLIP in addition to FIFO. The analytical model presented in [25] shows that IPS can improve the limitations of the FIFO mechanism by implementing the deterministic schemes such as the iSLIP and PIM. Although the IPS presented in [24,25] can be applied to the routers appropriately, none of them can be used in grid environments because of the different structures of the router systems and grid environments. In [24,25], it is assumed that all of the output queues can be accessed from all of the input ones, but in grid environments, the grid resources are belonged to the various administrative domains and cannot be accessed by all of the managers. In other words, the scheduling scheme in grid environments should be considered in two different levels; within the grid managers in the first level and within the grid resources in the second level.

Zhou and Beard [26] have introduced an adaptive probabilistic scheduling scheme for dealing with admission control in a cellular emergency network. The proposed method in [26] dynamically adjusts the scheduling probability regarding the arrival rates of the different classes of traffic. Moreover, the admission of emergency traffic and public originating traffic can be adjusted in this method. Rao and Huh [27] have proposed a probabilistic and adaptive job scheduling algorithm using system generated predictions for grid systems. The proposed algorithm first uses system-generated job execution time estimates without actually submitting jobs to the target resource. Then, this estimation is used to predict the job scheduling feasibility on the target system.

3. The system model

3.1. Grid environments

The grid computing environment considered in this paper is composed of a number of virtual organizations (VOs) in which share their own resources to shape a powerful computational system. The VOs which we are concerned can be small or large, short- or long-lived, and homogeneous or heterogeneous [3,11,12,16]. Individual VOs may be structured hierarchically from smaller systems and may overlap in membership. Each of these VOs is composed of several computational resources and a manager that manages the scheduling of the applications inside a VO. The manager of a VO is responsible for executing the tasks submitted to that VO and delivering the results of the execution to the corresponding user. Since VOs may overlap in membership [1,16], a manager belonging to a specific VO can administer the resources existing inside other VOs, and therefore a resource can be administered by more than one manager.

In grid environments, all of the resources are autonomous and unpredictable. They can be added to or taken away from the environment continuously [1,2]. Therefore, none of the managers have any control over the resources. Managers can only control the scheduling of the tasks and some administrative issues over member resources. Resource sharing, coordination, management and agreement negotiation among different VOs are to be conducted between managers. Actually, the relationship between VOs is made by connection through their managers. Fig. 1 shows the structure of the grid environment described above. As it is shown in Fig. 1, grid users submit their own tasks to one of the grid managers. The manager is responsible for execution of the submitted tasks. To execute a given task, the manager contacts its own resources to find a suitable resource to service the task. If the manager finds such resource, then it assigns the task to that resource, otherwise, the manger contacts other mangers within other VOs to find the suitable resource to service the submitted task. Since the goal of the scheduling in this paper is to minimize the overall mean response time of the tasks and total makespan of the grid environment, the grid managers should search the environment to find a path with minimum routing time (within managers) and a resource which can complete a given task in minimum possible time.

As shown in Fig. 1, a time label is assigned to each of the managers and resources. The time in a manager shows the mean time required to service a task in the manager. This service contains the entire search of the related domain to find the best suitable resource to execute the submitted task. The time associated to a resource shows the mean service time of the resource for executing a particular task, as well. Therefore, when a task is transferred between managers to reach a suitable resource for execution, the mean response times of the managers in which the task passed them are added to the overall response time of the task. Hence,

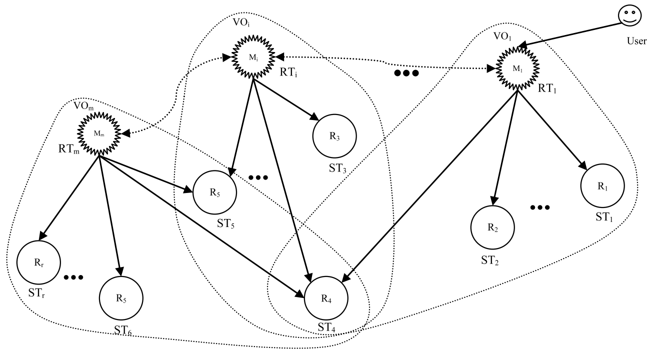

Fig. 1. A simple structure of the grid environment considered in this paper.

the mean response time of the task t assigned to the resource R_j, shown by $\overline{T}_{t_j}$, can be calculated by Eq. (1).

$$\overline{T}_{t_j} = \sum_{i;\ \text{task } t \text{ is passed through manager } M_i} \overline{RT}_i + \left(\overline{ST}_j + Ready_j\right) \tag{1}$$

where $\overline{RT}_i$ denotes the mean response time of the manager M_i, and $\overline{ST}_j$ and $Ready_j$ denote the mean service time and ready time of the resource R_j, respectively. It is necessary to mention that the ready time of a resource which show the time at which the resource is ready to execute the assigned task should be updated after each assignment. Therefore, the ready times are set to *zero* for all of the resources in beginning of the scheduling, and then they are updated after each assignment.

3.2. Assumptions

The following are some assumptions, which are used in the proposed scheduling method.

Assumption 1. The communication between various VOs is provided by communicating their managers. All of the managers can be fully connected to each other, but it is also possible to forbid some transitions.

Assumption 2. The communication links between the grid managers are bilaterally. In other words, if the manager M_i is connected to M_j, then M_i can send data to M_j and vice versa. This assumption is necessary to allow users to be in touch with grid environment via all of the managers. Actually, there is no central terminal to deliver user tasks, and therefore users can submit their own tasks to each of the managers.

Assumption 3. The communication links between the managers and resources are unilaterally. This means that, if the grid manager M_i is connected to the grid resource R_j, then only M_i can send data to R_j.

Assumption 4. There is no communication links between the grid resources. The relationship between the grid resources can only be provided by their managers.

Assumption 5. Tasks submitted to the managers have no data and control dependences on each other. Therefore, a grid resource can start the execution of the assigned task immediately after it gets the task data from the manager.

4. The proposed method

In this section, the proposed probabilistic scheduling method is presented. In order to present the method, some preliminaries of Markov chains [17,18] should be mentioned. Therefore, we explain

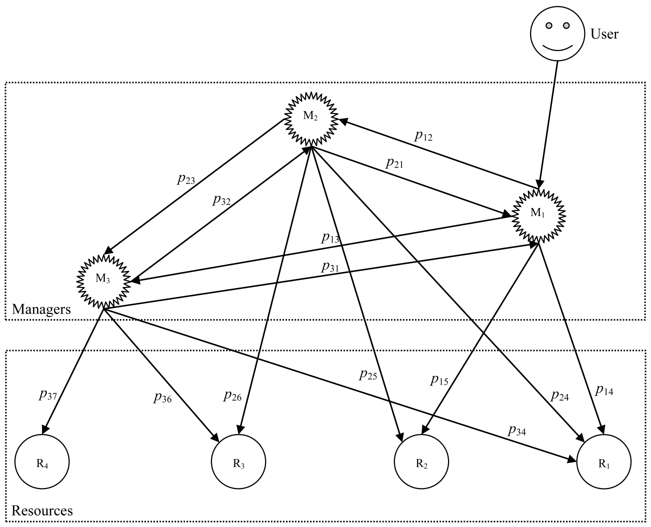

Fig. 2. A simple grid environment with three managers and four resources.

the required equations and formulas of Markov chains and then map the grid scheduling problem to the appropriate Markov chain. More details about the definitions and prove of the theorems can be found in Refs. [17,18,28].

Based on the concepts explained in Section 3, the connection among the managers and between a manger and its resources is very important, because it forms the topology of the state transition diagram representing the grid environment. As an example, consider a simple grid environment with three grid managers and four grid resources. Suppose all of the managers are connected to each other, but the connection between the managers and resources is different. Such that the manger M_1 is connected to the resources R_1 and R_2, the manager M_2 is connected to the resources R_1, R_2 and R_3, and finally the manager M_3 is connected to the resources R_1, R_3 and R_4. Fig. 2 shows this simple grid environment in which the managers and resources are depicted in different boxes.

For the grid environment described in Section 3, a matrix representing the connection between the managers and resources can be defined. This matrix, named connectivity matrix, shows the topology of the grid environment. In general, a connectivity matrix for a grid environment with m number of managers and r number of resources can be represented as Eq. (2)

$$\text{Conn} = \begin{array}{c|cccccc} & M_1 & \cdots & M_m & R_1 & \cdots & R_r \\ \hline M_1 & c_{11} & c_{12} & & & & c_{1(m+r)} \\ \cdots & & & & \cdots & & \\ M_m & c_{21} & c_{22} & & & & c_{2(m+r)} \\ R_1 & & \vdots & & \ddots & & \vdots \\ \cdots & & & & & & \\ R_r & c_{(m+r)1} & c_{(m+r)2} & & \cdots & & c_{(m+r)(m+r)} \end{array} \quad (2)$$

where $c_{ij} = \begin{cases} 1; \text{there is a connection between node } i \text{ and node } j, \\ 0; \text{otherwise,} \end{cases} \quad \forall i, j;\ 1 \le i, j \le (m + r)$.

In addition, the connection probability between managers and resources can be represented by a row matrix for each of the nodes (managers and resources) in the grid environment. The connection probability matrix for node i can be represented as Eq. (3).

$$\text{Prob}_i = \left[p_{i1}\, p_{i2}\, \cdots\, p_{i(m+r)}\right], \quad \forall i;\ 1 \le i \le (m+r) \quad (3)$$
$$0 \le p_{ij} \le 1, \quad \forall i, j;\ 1 \le i, j \le (m+r)$$
$$\sum_{j=1}^{(m+r)} p_{ij} = 1, \quad \forall i;\ 1 \le i \le (m+r)$$

where p_{ij} denote the probability that node i establish a connection with node j and sends a task to the node j. Actually, the aim of the proposed method is to find p_{ij} for all i and j in which the overall mean response time of the tasks is minimized.

Using Eqs. (2) and (3) for all of the nodes (managers and resources), one can obtain the probability matrix of the environment by multiplying Prob_i to the corresponding row in matrix *Conn*. Eq. (4) shows the probability matrix of a grid environment with m number of managers and r number of resources.

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1(m+r)} \\ p_{21} & p_{22} & & p_{2(m+r)} \\ \vdots & & \ddots & \vdots \\ p_{(m+r)1} & p_{(m+r)2} & \cdots & p_{(m+r)(m+r)} \end{bmatrix} \quad (4)$$
$$0 \le p_{ij} \le 1, \quad \forall i, j;\ 1 \le i, j \le (m+r)$$
$$\sum_{j=1}^{m+r} p_{ij} = 1, \quad \forall i;\ 1 \le i \le (m+r).$$

Since none of the nodes representing the grid managers have self-loop transitions, we can replace p_{ii} with *zero* for $1 \le i \le m$. Moreover, since there is no communication links among the grid resources, without loss of generality, we can suppose that there is a self-loop transition on each of the nodes representing the grid resources. Therefore, the entry p_{jj} of matrix P can be replaced with *one* for $m + 1 \le j \le m + r$. In our work, this modification is necessary to model the task scheduling process using Markov chains. Considering aforementioned modifications, matrix P shown in Eq. (4) can be rewritten as Eq. (5).

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1(m+r)} \\ p_{21} & p_{22} & & p_{2(m+r)} \\ \vdots & & \ddots & \vdots \\ p_{(m+r)1} & p_{(m+r)2} & \cdots & p_{(m+r)(m+r)} \end{bmatrix} \quad (5)$$
$$p_{ii} = 0, \quad \forall i;\ 1 \le i \le m$$
$$0 \le p_{ij} \le 1, \quad \forall i;\ 1 \le i \le m,\ \forall j;\ 1 \le j \le (m+r)$$
$$\sum_{j=1}^{m+r} p_{ij} = 1, \quad \forall i;\ 1 \le i \le m$$
$$p_{ii} = 1, \quad \forall i;\ (m+1) \le i \le (m+r)$$
$$p_{ij} = 0, \quad \forall i;\ (m+1) \le i \le (m+r),$$
$$\forall j;\ 1 \le j \le (m+r),\ i \ne j.$$

Matrix P shown in Eq. (5) specifies a state transition diagram (or finite directed graph), where state i in the graph is shown by a vertex, and a one-step transition from state i to state j is illustrated by an edge labeled by transition probability p_{ij}. Based on definition, a finite state discrete time Markov chain (DTMC) graphically can be shown by such state transition diagram [17]. Therefore, matrix P shows the one-step transition probability matrix of the $(m + r)$-state DTMC [17,18]. DTMCs can be used to effectively model and analyze the various performance measures of computer systems. DTMCs are categorized based on the classification of their constituent states [17,18,28].

Definition 1. A state s_i of a Markov chain is called *absorbing state* if it is impossible to leave it. In other words, state s_i is called absorbing state if and only if no other of the DTMC can be reached from it (i.e., $p_{ii} = 1$).

Definition 2. A Markov chain is *absorbing* if it has at least one absorbing state, and if from every state, it is possible to go to an absorbing state (not necessarily in one step). In an absorbing

Markov chain, a state which is not absorbing is called *transient* state.

Considering Eq. (5), Definitions 1 and 2, it can be expressed that DTMC representing the grid task scheduling is an absorbing DTMC. Actually, the nodes representing the grid resources form absorbing states in the related DTMC. Transition probability matrix of an arbitrary absorbing DTMC can be split into four sub-matrixes. If there are r absorbing states, representing grid resources, and m transient states, representing grid managers, the transition probability matrix of the DTMC can be shown using following canonical form:

$$P = \begin{array}{cc} & \begin{array}{cc}\text{TS} & \text{AbS}\end{array} \\ \begin{array}{c}\text{TS}\\ \text{AbS}\end{array} & \left(\begin{array}{c|c} Q & R \\ \hline 0 & I \end{array}\right)\end{array} \tag{6}$$

where I is an r-by-r identity matrix, 0 is an r-by-m zero matrix, R is a nonzero m-by-r matrix, and Q is an m-by-m matrix. The first m states are transient, shown with *TS*, and the last r states are absorbing which is demonstrated with *AbS*.

Matrix P shows the one-step transition probability between states of a DTMC. Repeatedly applying one-step transitions generalizes directly to n-step transition probabilities. More precisely, let $p_{ij}^{(n)}(k,l)$ denote to the probability that the DTMC transits from state i at time k to the state j at time l in $n = l - k$ steps. Eq. (7) shows this more accurately.

$$p_{ij}^{(n)}(k,l) = P\left(X_l = j | X_k = i\right), \quad 0 \le k \le l. \tag{7}$$

Now, the theorem of total probability applies for any given state i and any given time values k and l such that $\sum_j p_{ij}^{(n)}(k,l) = 1$, where $0 \le p_{ij}^{(n)}(k,l) \le 1$ [17].

Considering above definitions and concepts, following theorem can be written:

Theorem 1. *Let P be the transition probability matrix of a DTMC. The (i,j)th entry, $p_{ij}^{(n)}$, of the matrix P^n gives the probability that the DTMC, starting in state s_i, will be in state s_j after n steps.*

Now, using a standard matrix algebra argument, one can obtain P^n as following form:

$$P^n = \begin{pmatrix} Q^n & * \\ 0 & I \end{pmatrix} \tag{8}$$

where the asterisk shows the m-by-r matrix in the upper right hand corner of P^n. The form of P^n shows that the entries of Q^n present the probabilities of being in each of the transient states after n steps for each possible transient starting state [18]. Using *probability of absorption* theorem [18,28], it can be proved that in an arbitrary absorbing DTMC, the probability that the process will be absorbed is 1 (i.e., $Q^n \to 0$ as $n \to \infty$).

As mentioned before, the (i,j)th entry in Q^k denotes the probability of arriving in state s_j after just k steps, starting from state s_i. Therefore, we can define the matrix N as the inverse of matrix $I - Q$. It can be proved that the matrix N is equal to [18]:

$$N = I + Q + Q^2 + \cdots = \sum_{k=0}^{\infty} Q^k. \tag{9}$$

Definition 3. For an absorbing Markov chain MC, the matrix $N = (I-Q)^{-1}$ is called the *fundamental matrix* for MC. The entry n_{ij} of N shows the expected number of times that the process is in the transient state s_j if it is started in the transient state s_i.

If X_{ij} is a random variable denoting the number of times the state s_j is visited starting from s_i, before entering the absorbing state, then

$$E\left[X_{ij}\right] = n_{ij}, \quad 1 \le i,j \le m \tag{10}$$

where n_{ij} is the (i,j)th entry in N.

The visit number of a specific transient state is very useful for estimating QoS measures in a system such as the average time spent in the system, the overall reliability of the system and so forth. As an example, one can find the mean response time of a system by multiplying the average service times of nodes represented by transient states (s_j) to the expected number of times that the process is in the transient state s_j if it is started in the transient state s_i.

Having mean response time of the grid manager M_i and the mean service time of the grid resource R_j, the expected response time to a particular task can be estimated. To achieve this, the overall DTMC representing the grid environment should be split into small DTMCs (sub-DTMCs) with exactly one absorbing state. This means that, we can construct r distinct DTMCs from the primitive DTMC in which each of the new generated DTMCs consists of exactly one absorbing state representing its corresponding grid resource.

Now, the expected response time for each of the new generated DTMCs can be calculated. To do this, suppose task t is submitted to the grid manager M_k, and therefore this manager should find the appropriate path for scheduling the submitted task among the grid resources. To achieve this, M_k contacts other managers and its own resources to find a suitable resource for executing the submitted task. Suppose $\overline{RT}_i$ represents the mean response time of the manager M_i existing in the scheduling path and $\overline{ST}_j$ represents the mean service time of the resource R_j allocated to service the task t. Therefore, having Eq. (1) and considering the sub-DTMC consisting of resource R_j, the mean response time of the task t assigned to the resource R_j, shown by $\overline{T}_{t_j}$, can be calculated by Eq. (11).

$$\overline{T}_{t_j} = \sum_{i=1}^{m} \left(n_{ki} \times \overline{RT}_i\right) + \left(\overline{ST}_j + Ready_j\right) \tag{11}$$

where n_{ki} is the kth row of m-by-m matrix N.

Note that Eq. (11) can be written for each of the sub-DTMCs which can be generated inside the overall DTMC. In addition, the probability constraints, shown in Eq. (5), should be considered individually for each of the sub-DTMCs. In other words, only the probabilities of the edges existing in *sub-DTMC*$_j$, $1 \le j \le r$, will be taken into account in the related Eq. (11). Now, the aim is to find probabilities of the edges existing in *sub-DTMC*$_j$ such that the related mean response time, represented by $\overline{T}_{t_j}$, is minimized. To achieve this, considering Eq. (11) for each of the sub-DTMCs and its related probability constraints, a NLP problem will be defined for each of the sub-DTMCs. Solving the NLP problem for *sub-DTMC*$_j$ using methods such as branch and bound techniques [29], the probabilities of the edges existing in *sub-DTMC*$_j$ can be estimated. Since the aim of the proposed method is to find matrix P, we need to solve r numbers of these NLP problems to find the best solution. After that, the mean response time to the task t can be found by getting minimum of $\overline{T}_{t_j}$s. This minimization is shown in Eq. (12).

$$\overline{T}_t = \text{Min}\left(\overline{T}_{t_j}\right), \quad \forall j,\ 1 \le j \le r \tag{12}$$

where $\overline{T}_t$ denote the minimum mean response time of the task t submitted to the grid manager M_k. Knowing the minimum $\overline{T}_{t_j}$, which is equal to $\overline{T}_t$, and the related probabilities, one can shape the connection probability matrix of the grid environment for a given task. In some cases, a number of the entries in the probability matrix are left as unknown values. Actually, these values are not necessary for finding the optimal scheduling path. In these cases, mean response time of the scheduling, the path of the scheduling and the related probabilities can be found only using the computed values. Finally, after finding the optimal scheduling path, the suitable resource for executing the submitted task is determined. After determining the destination resource, the task is assigned to that resource, and then the ready time of the resource is updated. After that, the method can be applied to schedule next tasks.

There are two points which should be emphasized. First, there is a probability that $\overline{T}_t$ be equal to more than one $\overline{T}_{t_j}$. In this situation, one of the scheduling paths determined by one of the $\overline{T}_{t_j}$s can be chosen stochastically. Also, one can choose more than one path in each of the valid nodes with probability of (1/*number of the valid paths from that node*). Second, if $\overline{T}_t$ is equal to exactly one $\overline{T}_{t_j}$, our proposed method can find a deterministic scheduling path for the submitted task. In this case, the path of the scheduling and the resource allocated to the submitted task are specified deterministically. These two points are illustrated in the next section.

5. Illustrative examples

In order to show the applicability of the proposed method to the grid environments with different number of managers and resources, two simple grid environments with different topologies are presented, in this section. The first environment is composed of two managers and two resources with different processing speeds. In the second example, the grid environment depicted in Fig. 2 is considered and the probability matrix of this environment is calculated for different processing speeds of the managers and resources. In both examples, the parametric inverse of the fundamental matrix N is computed and the NLP problem obtained from optimization problem in each of the cases is solved using the LINGO optimization modeling software [30].

5.1. Example 1

Suppose there is a grid environment with two managers named M_1 and M_2 and two different resources named R_1 and R_2. Also, suppose the mean response times of the managers M_1 and M_2 and resources R_1 and R_2 are $\overline{RT}_1 = 1, \overline{RT}_2 = 2, \overline{ST}_1 = 9, \overline{ST}_2 = 7$ s per task, respectively. Moreover, it is assumed that there is no load on resources and both R_1 and R_2 can execute the assigned task immediately after getting its data from the related manager(s).

The connection probability matrix of this environment is given in Eq. (13).

$$P = \begin{bmatrix} 0 & p_{12} & p_{13} & 0 \\ p_{21} & 0 & p_{23} & p_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (13)$$

$$0 \le p_{12}, p_{13}, p_{21}, p_{23}, p_{24} \le 1$$
$$p_{12} + p_{13} = 1$$
$$p_{21} + p_{23} + p_{24} = 1.$$

Using matrix P shown in Eq. (13), the state transition diagram or DTMC of the environment can be depicted as Fig. 3.

The aim is to find matrix P such that the mean response time to the task t submitted to M_1 is minimized. To achieve this, the fundamental matrix N should be calculated for the DTMC shown in Fig. 3. Considering Definition 3, matrix N can be computed as Eq. (14).

$$N = \begin{bmatrix} (-1/(-1+p_{12}*p_{21})) & (-p_{12}/(-1+p_{12}*p_{21})) \\ (-p_{21}/(-1+p_{12}*p_{21})) & (-1/(-1+p_{12}*p_{21})) \end{bmatrix}. \qquad (14)$$

As shown in Fig. 3, two distinct absorbing DTMCs can be considered inside overall DTMC; DTMCs *Chain* 1 and *Chain* 2 consisting of

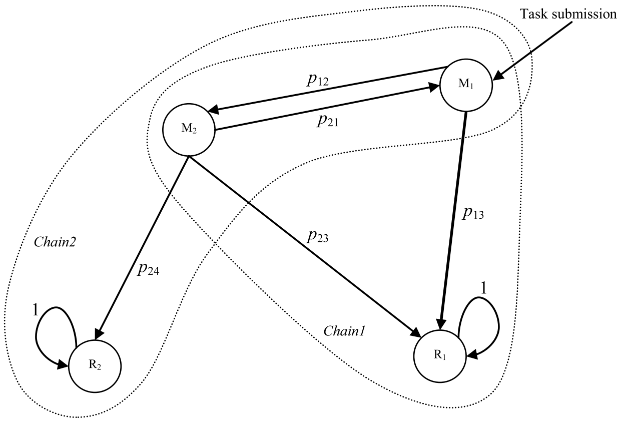

Fig. 3. DTMC representing the grid environment given in *Example* 1.

resources R_1 and R_2, respectively. Using Eq. (11), the mean response times of the task t can be computed when the task is assigned to R_1 (within *Chain* 1) and R_2 (within *Chain* 2). Eq. (15) shows these times.

$$\begin{aligned} \overline{T}_{t_1} &= \left((-1/(-1+p_{12}*p_{21})) \times \overline{RT}_1\right) \\ &\quad + \left((-p_{12}/(-1+p_{12}*p_{21})) \times \overline{RT}_2\right) + \left(\overline{ST}_1 + 0\right) \qquad (15) \\ \overline{T}_{t_2} &= \left((-1/(-1+p_{12}*p_{21})) \times \overline{RT}_1\right) \\ &\quad + \left((-p_{12}/(-1+p_{12}*p_{21})) \times \overline{RT}_2\right) + \left(\overline{ST}_2 + 0\right). \end{aligned}$$

The corresponding probability constraints among the entries of matrix P, shown in Eq. (13), are listed below. These constraints are necessary for solving Eq. (15).

$$\begin{aligned} &\text{Chain 1: for solving } \overline{T}_{t_1} \qquad (16) \\ &0 \le p_{12}, p_{13}, p_{21}, p_{23} \le 1 \\ &p_{12} + p_{13} = 1 \\ &p_{21} + p_{23} = 1 \\ &\text{Chain 2:for solving } \overline{T}_{t_2} \\ &0 \le p_{12}, p_{21}, p_{24} \le 1 \\ &p_{12} = 1 \\ &p_{21} + p_{24} = 1. \end{aligned}$$

Using Eq. (12) and considering Eqs. (15) and (16), the NLP problem to find the minimum mean response time of the task t can be written as Eq. (17).

$$\begin{aligned} &\overline{T}_t = \text{Min}\left(\overline{T}_{t_1}, \overline{T}_{t_2}\right) \qquad (17) \\ &\overline{T}_{t_1} = \left((-1/(-1+p_{12}*p_{21})) \times \overline{RT}_1\right) \\ &\qquad + \left((-p_{12}/(-1+p_{12}*p_{21})) \times \overline{RT}_2\right) + \overline{ST}_1 \\ &0 \le p_{12}, p_{13}, p_{21}, p_{23} \le 1 \\ &p_{12} + p_{13} = 1 \\ &p_{21} + p_{23} = 1 \\ &\overline{T}_{t_2} = \left((-1/(-1+p_{12}*p_{21})) \times \overline{RT}_1\right) \\ &\qquad + \left((-p_{12}/(-1+p_{12}*p_{21})) \times \overline{RT}_2\right) + \overline{ST}_2 \\ &0 \le p_{12}, p_{21}, p_{24} \le 1 \\ &p_{12} = 1 \\ &p_{21} + p_{24} = 1. \end{aligned}$$

As it is obvious in Eq. (17), p_{12} is equal to one, and therefore; p_{21} and p_{24} can be found easily. Nevertheless, this is only a simple example and in general, solving the optimization problem and finding the related probabilities are not easy works. Therefore, in order to solve the NLP problems, LINGO optimization engine is used. Using

Table 1
Results obtained from solving system (17).

Chain 1		Chain 2	
Local optimal solution found		Local optimal solution found	
Objective value:	8.00000	Objective value:	10.00000
Model class:	NLP	Model class:	NLP
Variable	Value	Variable	Value
NUMBER_P	4.000000	NUMBER_P	3.000000
NUMBER_T	4.000000	NUMBER_T	4.000000
MEAN_RESPNSE_TIME	8.00000	MEAN_RESPNSE_TIME	10.00000
p_{12}	0.000000	p_{12}	1.000000
p_{13}	1.000000	p_{21}	0.000000
p_{21}	0.0695951	p_{24}	1.000000
p_{23}	0.9304048	RT_1	1.000000
RT_1	1.000000	RT_2	2.000000
RT_2	2.000000	ST_1	9.000000
ST_1	9.000000	ST_2	7.000000
ST_2	7.000000		

LINGO, values of p_{12}, p_{13}, p_{21} and p_{23} shown in *Chain* 1 and p_{12}, p_{21} and p_{24} shown in *Chain* 2 can be calculated as Table 1.

In Table 1, *Objective value* denotes the minimum mean response time obtained from solving the related NLP problem in each of the sub-DTMCs. Also, *NUMBER_P* denotes the number of the probabilities which should be found during solving the problem, and *NUMBER_T* represents the number of the mean response and service times related to the managers and resources, respectively. As shown in Table 1, the values computed for $\overline{T}_{t_1}$ and $\overline{T}_{t_2}$ are 8.0 and 10.0, respectively. Therefore, the value of $\overline{T}_t$ will be equal to $\overline{T}_{t_1}$ which is 8.0. Using this fact, the entries of matrix P, shown in Eq. (13), can be calculated considering *Chain* 1 probability constraints. Eq. (18) shows matrix P.

$$P = \begin{bmatrix} 0 & \mathbf{0} & \mathbf{1} & 0 \\ \mathbf{0.07} & 0 & \mathbf{0.93} & \mathbf{UN} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{18}$$

As it is shown in Eq. (18), the p_{12} and p_{13} entries of matrix P are equal to 0 and 1, respectively. Considering Fig. 3, it can be concluded that the task t submitted to the manager M_1 will be assigned to the resource R_1, and therefore the mean response time of the task will be equal to 8.0 s. The path of the scheduling is also shown by bold arrow in Fig. 3. Therefore, in this case, our proposed scheduling method can find the optimal scheduling deterministically.

After finding the scheduling path and the destination resource, the task is assigned to the resource and ready times of the resources are updated. In this example, the ready time of the resource R_1 will be the sum of its previous ready time and its service time $(0 + 7.0 = 7.0)$. Therefore, in the next step, the new ready times will be used instead of the previous ones. It is necessary to mention that the value of the entry p_{24} in matrix P, shown in Eq. (18), is left as *unknown value* (*UN*). This value is not required for finding the optimal scheduling because the edge related to p_{24} does not belong to the optimal scheduling path.

5.2. Example 2

Consider the grid environment described in Section 4. This environment consists of three managers named M_1, M_2 and M_3 and four grid resources named R_1, R_2, R_3, and R_4. The mean response times of the managers M_1 to M_3 and the mean service times of the resources R_1 to R_4 are supposed to be $\overline{RT}_1 = 1.5$, $\overline{RT}_2 = 1$, $\overline{RT}_3 = 2$, $\overline{ST}_1 = 3$, $\overline{ST}_2 = 4$, $\overline{ST}_3 = 1.5$ and $\overline{ST}_4 = 0.5$ s per task, respectively. This grid environment is depicted in Fig. 2 and the connection probability matrix of the environment is given in Eq. (19).

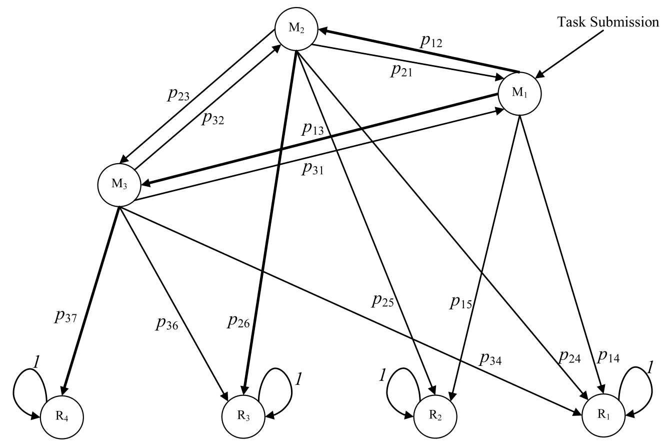

Fig. 4. DTMC representing the grid environment given in *Example* 2.

$$P = \begin{bmatrix} 0 & p_{12} & p_{13} & p_{14} & p_{15} & 0 & 0 \\ p_{21} & 0 & p_{23} & p_{24} & p_{25} & p_{26} & p_{27} \\ p_{31} & p_{32} & 0 & p_{34} & 0 & p_{36} & p_{37} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

$$0 \le p_{ij} \le 1, \quad \forall p_{ij};\ p_{ij} \in P$$

$$\sum_{j=2}^{5} p_{1j} = 1$$

$$\sum_{j=1,\ j\neq 2}^{7} p_{2j} = 1$$

$$\sum_{j=1,\ j\neq 3,5}^{7} p_{3j} = 1.$$

Considering matrix P, shown in Eq. (19), the related DTMC can be depicted as Fig. 4.

In order to find matrix P, the fundamental matrix N should be computed for the DTMC shown in Fig. 4. Applying Eq. (11) to the fundamental matrix N for each of the sub-DTMCs, the possible mean response times for the task t can be calculated when t is assigned to the resources R_1 to R_4. For the sake of brevity, the fundamental matrix, formulas related to the mean response times of the sub-DTMCs and their probability constraints are not presented here. Solving the NLP problem defined by minimizing $\overline{T}_{t_i}$ for $1 \le i \le 4$, the required entries of the transition probability matrix can be calculated. Results obtained from solving the minimization problems using LINGO optimization engine are summarized in Table 2.

As shown in Table 2, the values computed for $\overline{T}_{t_1}$, $\overline{T}_{t_2}$, $\overline{T}_{t_3}$ and $\overline{T}_{t_4}$ are 4.5, 5.5, 4 and 4, respectively. Therefore, the value of $\overline{T}_t$ will be equal to $\overline{T}_{t_3}$ and $\overline{T}_{t_4}$ which are 4. Considering the probabilities obtained from minimizing $\overline{T}_{t_3}$ and $\overline{T}_{t_4}$, two possible alternatives for matrix P, shown in Eq. (19), can be conceived as Eqs. (20) and (21).

$$P = \begin{bmatrix} 0 & \mathbf{1} & \mathbf{0} & \mathbf{UN} & \mathbf{UN} & 0 & 0 \\ \mathbf{0} & 0 & \mathbf{0} & \mathbf{UN} & \mathbf{UN} & \mathbf{1} & \mathbf{UN} \\ \mathbf{0} & \mathbf{0} & 0 & \mathbf{UN} & 0 & \mathbf{1} & \mathbf{UN} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

Table 2
Results obtained from optimization problem related to *Example* 2.

sub-DTMC1		sub-DTMC2	
Local optimal solution found		Local optimal solution found	
Objective value:	4.500000	Objective value:	5.50000
Model Class:	NLP	Model Class:	NLP
Variable	Value	Variable	Value
NUMBER_P	9.000000	NUMBER_P	8.000000
NUMBER_T	7.000000	NUMBER_T	7.000000
MEAN_RESPNSE_TIME	4.500000	MEAN_RESPNSE_TIME	5.500000
p_{12}	0.000000	p_{12}	0.000000
p_{13}	0.000000	p_{13}	0.000000
p_{21}	0.000000	p_{21}	0.000000
p_{23}	0.000000	p_{23}	0.000000
p_{31}	0.000000	p_{31}	1.000000
p_{32}	0.000000	p_{32}	0.000000
p_{14}	1.000000	p_{15}	1.000000
p_{24}	1.000000	p_{25}	1.000000
p_{34}	1.000000	RT_1	1.500000
RT_1	1.500000	RT_2	1.000000
RT_2	1.000000	RT_3	2.000000
RT_3	2.000000	ST_1	3.000000
ST_1	3.000000	ST_2	4.000000
ST_2	4.000000	ST_3	1.500000
ST_3	1.500000	ST_4	0.500000
ST_4	0.500000		
sub-DTMC3		sub-DTMC4	
Local optimal solution found		Local optimal solution found	
Objective value:	4.000000	Objective value:	4.000000
Model Class:	NLP	Model Class:	NLP
Variable	Value	Variable	Value
NUMBER_P	8.000000	NUMBER_P	7.000000
NUMBER_T	7.000000	NUMBER_T	7.000000
MEAN_RESPNSE_TIME	4.000000	MEAN_RESPNSE_TIME	4.000000
p_{12}	1.000000	p_{12}	0.000000
p_{13}	0.000000	p_{13}	1.000000
p_{21}	0.000000	p_{21}	0.000000
p_{23}	0.000000	p_{23}	1.000000
p_{31}	0.000000	p_{31}	0.000000
p_{32}	0.000000	p_{32}	0.000000
p_{26}	1.000000	p_{37}	1.000000
p_{36}	1.000000	RT_1	1.500000
RT_1	1.500000	RT_2	1.000000
RT_2	1.000000	RT_3	2.000000
RT_3	2.000000	ST_1	3.000000
ST_1	3.000000	ST_2	4.000000
ST_2	4.000000	ST_3	1.500000
ST_3	1.500000	ST_4	0.500000
ST_4	0.500000		

$$P = \begin{bmatrix} 0 & \mathbf{0} & \mathbf{1} & \mathbf{UN} & \mathbf{UN} & 0 & 0 \\ \mathbf{0} & 0 & \mathbf{UN} & \mathbf{UN} & \mathbf{UN} & \mathbf{UN} & \mathbf{UN} \\ \mathbf{0} & \mathbf{0} & 0 & \mathbf{UN} & 0 & \mathbf{UN} & \mathbf{1} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{21}$$

Therefore, there are two optimal scheduling in this problem. One scheduling path is $M_1 \xrightarrow{p_{12}=1} M_2 \xrightarrow{p_{26}=1} R_3$ and the other one is $M_1 \xrightarrow{p_{13}=1} M_3 \xrightarrow{p_{37}=1} R_4$. Both of these scheduling paths result the same mean response time for the submitted task. Therefore, the grid scheduler can select one of them stochastically. Moreover, the scheduler can use the combination of them in which each of the paths determined by Eqs. (20) and (21) are selected with suitable probabilities. One possible combinatorial probability matrix for this problem can be written as Eq. (22) in which the manager M_1 schedules the task to the managers M_2 and M_3 with the same probability equal to 0.5. Afterward, the managers M_2 and M_3 schedule the task to the resources R_3 and R_4, respectively. Both scheduling paths are shown by bold arrows in Fig. 4.

$$P = \begin{bmatrix} 0 & \mathbf{0.5} & \mathbf{0.5} & \mathbf{UN} & \mathbf{UN} & 0 & 0 \\ \mathbf{UN} & 0 & \mathbf{UN} & \mathbf{UN} & \mathbf{UN} & \mathbf{1} & \mathbf{UN} \\ \mathbf{UN} & \mathbf{UN} & 0 & \mathbf{UN} & 0 & \mathbf{UN} & \mathbf{1} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{22}$$

It should be mentioned that the numbers of the path probabilities are different in sub-DTMCs. As it can be seen in Table 2, the values of *NUMBER_P* in *sub-DTMC*1,*sub-DTMC* 2,*sub-DTMC*3 and *sub-DTMC*4 are equal to 9, 8, 8 and 7, respectively. Similar to the *Example* 1, after finding the suitable scheduling path and the target resource, the task is assigned to the resource and the ready times of the resources are updated.

6. Performance evaluation

In this section, performance of the proposed method is compared to some similar scheduling algorithms regarding two parameters: *overall mean response time (OMRT) difference* and *total makespan*. To do this, in the first step, five scheduling algorithms which are used as benchmarks in many research works are introduced, and then the comparison parameters are described. Since our proposed method schedules a task to the appropriate resource immediately after submitting the task to grid managers, the proposed method falls into immediate mode scheduling scheme [31,32]. Therefore, five immediate mode scheduling algorithms; opportunistic load balancing (OLB), minimum execution time (MET), minimum completion time (MCT), switching algorithm (SA), *k*-percent best (*k*PB) [31–35], are selected to be compared with our proposed scheduling method. The short introduction about each of the mentioned heuristics is presented in below [32–35].

- *OLB*: This algorithm assigns each task to the earliest idle resource without any consideration about the execution time of the task on the resource. If two or more resources are idle, then a resource is selected arbitrarily. The intuition behind OLB is to keep all resources as busy as possible. One advantage of OLB is its simplicity, but because OLB does not take the task execution times into account, the resulting schedule is not optimal.
- *MET*: This algorithm assigns each task to a resource that results in the least execution time for that task, regardless of that machine's availability. As a task arrives, all the resources in the environment are examined to determine the resource that gives the minimum execution time for the task. Therefore, the motivation behind MET is to give each task to its best resource. But, allocating task without considering resource availability results in load imbalance on grid resources.
- *MCT*: This algorithm assigns a task to the resource yielding the earliest completion time (ready time of resource + task execution time on that resource) for that task. When a task arrives in the environment, all available resources are examined to determine the resource that yields the smallest completion time for the task. In MCT, a task could be assigned to a resource that does not have the smallest execution time for that task. The intuition behind MCT is to combine the advantages of OLB and MET, while avoiding their drawbacks. This method is also known as Fast Greedy, originally proposed for SmartNet system.
- *SA*: The MET method has a potential drawback in that it can lead to load imbalance across resources by assigning many more tasks to some resources than to the others since it blindly looks at execution times of the tasks without considering the ready time of the resources. On the other hand, the

MCT heuristic assigns tasks to resources to achieve earliest completion time thereby ensuring load balance but does not necessarily minimize the execution times of the tasks. The SA tries to overcome some limitations of MET and MCT methods by combining their best features. Here, the idea is to first use the MCT until a threshold of balance is obtained followed by MET which creates the load imbalance by assigning tasks on faster resources. More precisely, let $r_{\max}$ be the maximum ready time and $r_{\min}$ be the minimum ready time; the load balancing factor is then $r = r_{\min}/r_{\max}$ which takes values in range [0, 1]. It is obvious that for $r = 1$ we have a perfect load balancing and if $r = 0$, then there exists at least one idle resource. Further, two threshold values; r_l and r_h, $0 \leq r_l \leq r_h = 1$ are used to control the order in which MCT and MET are applied. Initially, r is set to *zero* so that SA starts allocating tasks according to MCT until r becomes greater than r_h; after that, MET is activated so that r becomes smaller than r_l and a new cycle starts again until all tasks are allocated.

- *k*PB: This method also tries to combine the best features of MCT and MET simultaneously instead of cyclic manner. In this method, only k percentage of best resources, considering their service times, are chosen while assigning the tasks. For a particular task, a resource which gives minimum completion time is selected from of the k percent best resources instead of all possible resources. It should be noted that for $k = 100$, *k*PB behaves as MCT and for $k = (100/\text{total number of resources})$ it acts as MET.

After introducing the benchmark algorithms, comparison parameters should be described. As mentioned above, two comparison parameters; *OMRT difference* and *total makespan* are used to compare the algorithms with each other. In order to calculate the first parameter, mean response time of all the tasks submitted to the grid environment are firstly determined in each of the algorithms. Then, differences between the mean response times of the comparable tasks in the proposed method and the benchmark algorithms are computed and summed. After that, the number obtained from previous step is divided by the total number of tasks submitted to the environment to give our first comparison parameter. This parameter which is calculated for each of the benchmark algorithms can be used as a measure to compare the algorithms with respect to the resulting mean response times for all of the tasks. The second comparison parameter, total makespan, can be computed as the maximum time among all resources' ready times. In other words, after each assignment, the ready times of the resources are updated. After assigning all of the tasks to the resources, the maximum ready time shows the total makespan of the environment. Actually, this is the time that the grid environment finishes the execution of all the tasks submitted to the environment. Using these parameters, the performance of the proposed method can be properly compared to the other similar scheduling methods, because the first parameter compares the methods considering the mean response time of single tasks, and the second one compares the methods regarding the total completion time of all the tasks.

In order to compare the proposed method with the others, two different case studies are considered and the results obtained from simulations are reported. In the first case study, the simple grid environment explained in *example* 2 is considered and in the second one, the most realistic grid environment with more number of grid managers and resources is simulated.

6.1. Case study 1

Consider the grid environment described in Section 5.2 with the same number of grid managers and resources. Moreover, the

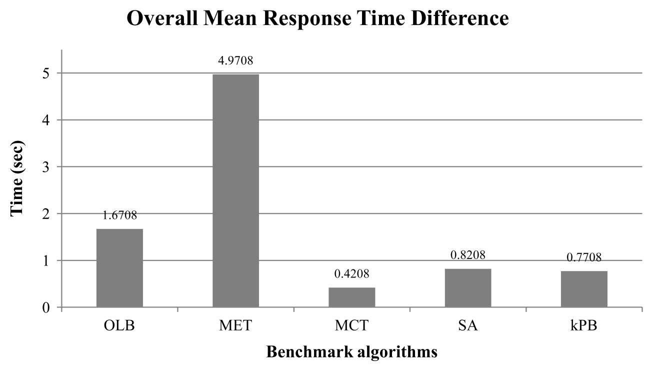

Fig. 5. OMRT differences obtained from benchmark algorithms in *case study* 1.

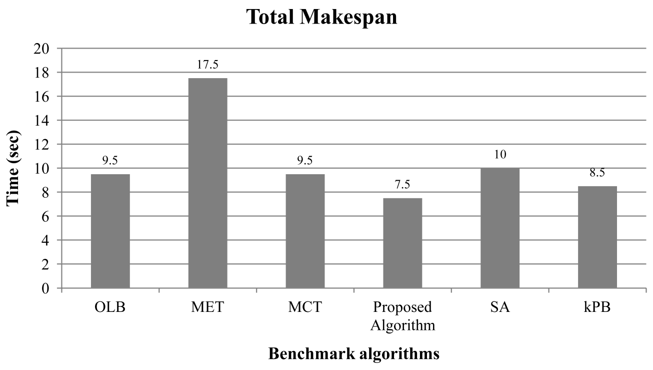

Fig. 6. Total makespan of the grid environment considered in *case study* 1.

topology of the environment and mean response and service times of the managers and resources are the same. It is supposed that the number of tasks submitted to the manger M_1 is 10, and the comparison parameters are calculated after execution of all of the tasks. Also, the r_l and r_h parameters in SA and k factor in *k*PB algorithm have been set to 0.1, 0.3 and 0.5, respectively. It should be mentioned that these values can be found in some related references, but generally, one could use other values for these parameters and compare the new results with our reported ones. The aforementioned values are approximately the best ones in which the results obtained from applying them are very reasonable and acceptable.

Fig. 5 compares the proposed algorithm with the mentioned benchmarks regarding the first comparison parameter, OMRT difference. As it can be seen in Fig. 5, the OMRT obtained from the proposed algorithm is less than the OMRTs of the other algorithms, because the differences between OMRT of the benchmarks and the proposed algorithm are positive. Moreover, considering Fig. 5, it can be concluded that the MCT and MET heuristics are respectively the closest and farthest methods from our proposed one regarding the first comparison parameter.

Fig. 6 shows the total makespan of the assumed grid environment after assigning all of 10 tasks to the resources. As it is obvious in Fig. 6, the total makespan of the environment obtained from applying the proposed method to the environment is less than other benchmarks' makespans. Considering Figs. 5 and 6, it is concluded that although the OMRT of the *k*PB algorithm is higher than MCT heuristic, the total makespan obtained from applying *k*PB is less than MCT's makespan.

As it is mentioned above, the grid environment in this case study is only a simple grid with small number of managers and resources. Furthermore, the number of the tasks submitted to the

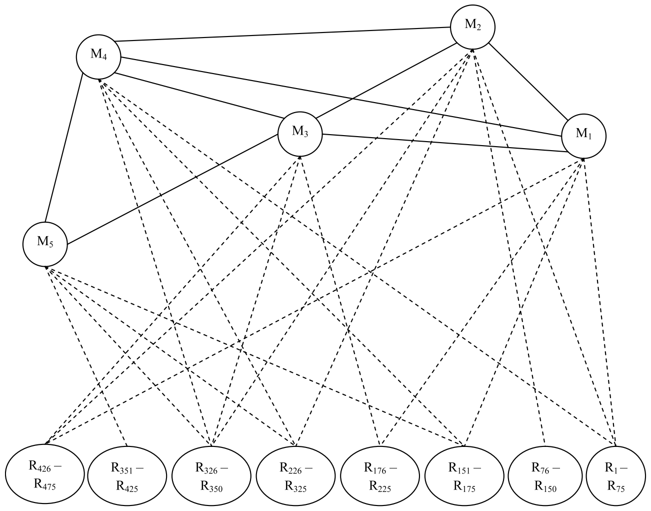

Fig. 7. State diagram representing the grid environment given in *case study* 2.

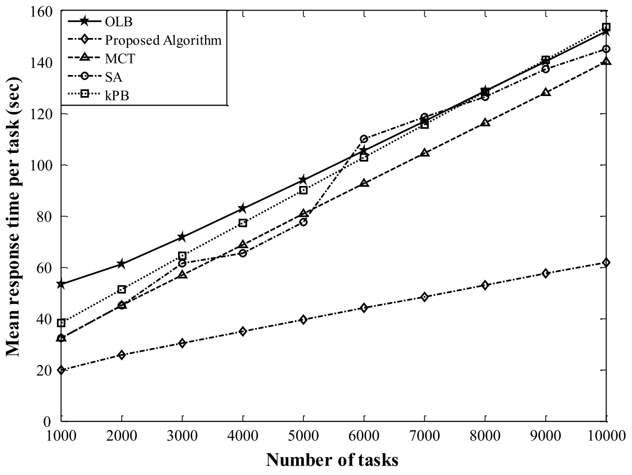

Fig. 8. OMRTs of the algorithms in *case study* 2.

environment is equal to 10, which is very small in comparison with real grids. For this reason, in the following case study, a most realistic grid is considered.

6.2. Case study 2

In order to study a most practical and realistic grid, a grid environment with 5 managers and 475 resources is considered. The number of the resources and virtual organizations in the grid discussed in this case study is the same as AuverGrid system [36]. Fig. 7 shows the state diagram of this environment in which the connections among managers are shown by solid arrows and the connections between managers and resources are shown by dashed arrows. It should be mentioned that the topology of the grid considered in this case study is different from the AuverGrid's topology, because we want to study the resource overlapping problem among different administrative domains, which is not a case in AuverGrid. For the sake of brevity, all the 475 grid resources are grouped within eight circles. Each circle shows a collection of resources which are connected to the same managers. As an example, as shown in Fig. 7, there are three dashed arrows between circles representing managers M_1, M_2, and M_4 and the circle representing the first group of resources named R_1–R_{75}. This demonstrates that the managers M_1, M_2, and M_4 are connected to all of the resources R_1–R_{75}.

All of the settings related to the benchmark algorithms in *case study* 2 are the same as those are in *case study* 1. Only, there are two main differences between these two case studies:

1. As mentioned in the *case study* 1, all of the tasks submitted to the grid environment are delivered to the manager M_1. But, in this case study, it is supposed that all of the five managers can receive tasks from the users. This assumption is made to provide grid users with distributed terminals to submit their own tasks.
2. The other difference between *case study* 2 and *case study* 1 is the number of the benchmark algorithms. In *case study* 2, the MET heuristic has been removed from the set of benchmarks because of its higher OMRT and makespan. As it can be seen in Figs. 5 *and* 6, the OMRT and the total makespan obtained from applying MET heuristic to the grid environment assumed in *case study* 1 are significantly higher than those are in other benchmarks. Consequently, since the number of tasks considered in *case study* 2 are very larger than *case study* 1's tasks, the comparison parameters (OMRT and total makespan) obtained from MET heuristic will be very different from the other benchmarks' parameters, so they cannot be appropriately justified in a chart. Therefore, the MET heuristic is eliminated from the benchmark algorithms and the proposed method is compared to the four remaining benchmarks, OLB, MCT, SA and *k*PB, in this case study.

In order to compare the proposed method to the aforementioned benchmarks, two different experiments are investigated in the *case study* 2. In the first experiment, the OMRTs of the algorithms are compared to each other. To do this, the number of tasks submitted to the environment is varied from 1000 to 10 000 in which 20% of all the tasks are delivered to each of the five managers, in each case. For example, when the number of submitted tasks is 1000, the tasks T_1 to T_{200}, T_{201} to T_{400}, T_{401} to T_{600}, T_{601} to T_{800} and T_{801} to T_{1000} are submitted to the managers M_1, M_2, M_3, M_4 and M_5, respectively. Fig. 8 displays the OMRTs obtained from the first experiment. As shown in Fig. 8, OMRT achieved from the proposed method is the lowest one among all of the obtained results. Moreover, although the OMRT of the MCT heuristic is higher than the proposed method's OMRT, the MCT shows relatively low OMRT compared to the other benchmark algorithms. Furthermore, as it can be seen in Fig. 8, the OMRT of SA do not follow a specific pattern and differ when the number of tasks is varied. This is because SA alternatively changes its own strategy to match with MCT or MET considering values of r_l and r_h. Since the number of tasks effects on the values of r_l and r_h, SA shows various OMRT for each number of tasks and does not obey a specific model for all number of tasks. But, generally, it can be stated that the SA's OMRT is always higher than the OMRT of the proposed method.

In the second experiment, the OMRT and total makespan obtained from applying the mentioned algorithms to the grid environment assumed in *case study* 2 are compared to each other, when the number of tasks is 400 000. This experiment can demonstrate the most realistic results because the number of tasks is very close to the real environments. Fig. 9 shows the OMRT differences between the benchmark algorithms and the proposed method. As it can be seen in Fig. 9, the OMRT obtained from the proposed method is less than the OMRTs of the other algorithms. This result is the same as one obtained from investigating Fig. 5. Hence, it can be concluded that the proposed method shows the minimum OMRT compared to the other algorithms in both light and heavy load of the system.

Fig. 10 shows the total makespan of the environment after assigning all of 400 000 tasks to the grid resources. As shown in Fig. 10, the total makespan of the environment obtained from applying the proposed method is less than other benchmarks' makespans. Moreover, one can see that the total makespan

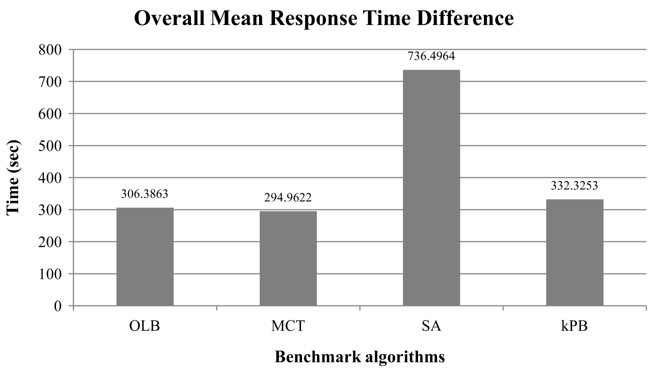

Fig. 9. OMRT differences obtained from benchmark algorithms in *case study* 2.

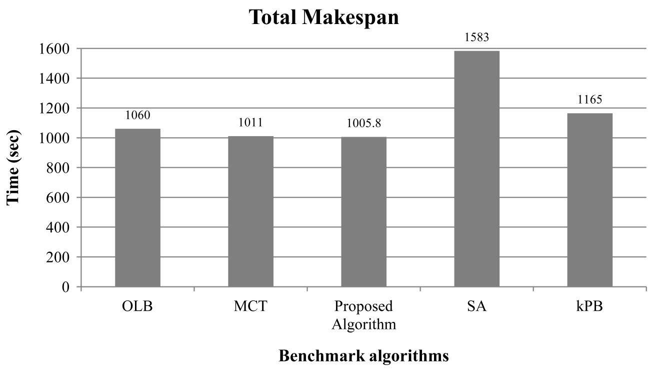

Fig. 10. Total makespan of the grid environment investigated in *case study* 2.

resulted from MCT heuristic is also an acceptable value, but the makespan of the proposed method is less than this value, too.

7. Conclusion and future work

The mean response time of a task is one of the most important user-oriented QoS measures in grid environments. Minimizing the mean response time of the tasks submitted to the grid environment can be useful to achieve high performance computing within grid environment. Furthermore, the total makespan of the grid is known as one of the most important system-oriented performance measures in which minimizing it can help the system to seem more effective and useful. In order to minimize the mean response time of the tasks and total makespan of the grids, appropriate task scheduling algorithms should be applied by the grid managers to optimally dispatch tasks to the available resources. In this paper, a new probabilistic scheduling method is presented to find the best suitable scheduling path within grid environments. Our proposed method can be run on each of the managers distributed through the environment and find the most suitable scheduling path. The proposed method uses DTMCs to model the task scheduling process within grids, and then applies quantitative analysis to the obtained DTMC. Considering the probability constraints on each of the absorbing DTMCs existing inside the overall DTMC, a NLP problem will be defined for each of the sub-DTMCs. Consequently, the minimum mean response time of a task and its corresponding scheduling path can be found by solving the NLP problems.

There are numbers of research issues remaining open for future work. The following are some ideas that can be used for further research in this area:

- *Using other QoS measures (instead of mean response time) as the target of the scheduling algorithm*: Taking into account other QoS measures (e.g. reliability of task execution, availability of the resources, cost of scheduling and so forth) may result new scheduling algorithms in grid environments. In addition, combining two or more QoS measures to propose a general scheduling method is an interesting issue in this field of research.
- *Applying a mechanism to select critical and impatient tasks from the list of the submitted tasks to be rapidly executed on grid resources*: This idea can be implemented by modeling the task scheduling using queuing networks. Afterward, applying the various scheduling mechanisms (e.g. EDF discipline) to the queues existing in the network and analyzing the network in the steady state, the throughput of the environment and performance of the scheduling can be assessed.
- *Taking into account the data and control dependences between the tasks submitted to the environment*: In our proposed scheduling method, none of the tasks have dependency to each other. But, in more realistic grid models, it is assumed that the tasks are dependent on each other and the execution of a particular task can be started only after execution of its preceding tasks.
- *Proposing a general model to consider the local tasks*: Generally, local tasks are submitted to the grid resources by local users in each of the administrative domains. In most cases, local tasks have higher priority than grid ones. Considering local tasks in the scheduling algorithms may result most applicable scheduling in grid environments.

References

[1] I. Foster, C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, 2nd ed., Morgan Kaufmann, 2004.
[2] M. Li, M. Baker, The Grid Core Technologies, 1st ed., John Wiley and Sons, 2005.
[3] K. Krauter, R. Buyya, M. Maheswaran, A taxonomy and survey of grid resource management systems for distributed computing, Journal of Software: Practice and Experience 32 (2) (2002) 135–164.
[4] M.A. Azgomi, R. Entezari-Maleki, Task scheduling modelling and reliability evaluation of grid services using coloured Petri nets, Future Generation Computer Systems 26 (8) (2010) 1141–1150.
[5] R. Plestys, G. Vilutis, D. Sandonavicius, The measurement of grid QoS parameters, in: The 29th International Conference on Information Technology Interfaces, Croatia, June 25–28, 2007, pp. 703–707.
[6] S. Parsa, R. Entezari-Maleki, Task dispatching approach to reduce the number of waiting tasks in grid environments, The Journal of Supercomputing (2010), in press (doi:10.1007/s11227-010-0448-5).
[7] P.M. Broadwell, Response time as a performability metric for online services, UC Berkeley Computer Science Technical Report, UCB CSD-04-1324, May 2004.
[8] X. Deng, Y. Zhang, Minimizing mean response time in batch processing system, in: The 5th Annual International Conference on Computing and Combinatorics, Japan, July 26–28, 1999, pp. 231–240.
[9] H. Li, R. Buyya, Model-based simulation and performance evaluation of grid scheduling strategies, Future Generation Computer Systems 25 (4) (2009) 460–465.
[10] E. Elmroth, J. Tordsson, Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions, Future Generation Computer Systems 24 (6) (2008) 585–593.
[11] B. Yagoubi, Y. Slimani, Task load balancing strategy for grid computing, Journal of Computer Science 3 (3) (2007) 186–194.
[12] A. Afzal, A.S. McGough, J. Darlington, Capacity planning and scheduling in grid computing environment, Future Generation Computer Systems 24 (5) (2008) 404–414.
[13] R. Entezari-Maleki, A. Movaghar, A genetic-based scheduling algorithm to minimize the makespan of the grid applications, in: T. Kim, et al. (Eds.), Grid and Distributed Computing, Control and Automation, in: Communications in Computer and Information Science (CCIS), vol. 121, Springer, 2010, pp. 22–31.
[14] E.J. Byun, et al., MJSA: Markov job scheduler based on availability in desktop grid computing environment, Future Generation Computer Systems 23 (4) (2007) 616–622.
[15] Z. Farkas, P. Kacsuk, Evaluation of hierarchical desktop grid scheduling algorithms, Future Generation Computer Systems (2011), in press (doi:10.1016/j.future.2010.12.013).
[16] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, International Journal of High Performance Computing Applications 15 (3) (2001) 200–222.
[17] G. Bolch, S. Greiner, H.D. Meer, K.S. Trivedi, Queueing Networks and Markov Chains, 2nd ed., John Wiley and Sons, 2006.
[18] C.M. Grinstead, J.L. Snell, Introduction to Probability, 2nd ed., American Mathematical Society, 1997.
[19] A. Bestavros, D. Spartiotis, Probabilistic job scheduling for distributed real-time applications, in: The 1st IEEE Workshop on Real-Time Applications, USA, May 13–14, 1993, pp. 97–101.

[20] L. Anand, D. Ghose, V. Mani, Probabilistic load scheduling with priorities in distributed computing systems, Computers and Operations Research 25 (10) (1998) 839–856.
[21] Y. Jiang, C.K. Tham, C.C. Ko, A probabilistic priority scheduling discipline for high speed networks, in: The IEEE Workshop on High Performance Switching and Routing, USA, May 29–31, 2001, pp. 1–5.
[22] Y. Jiang, C.K. Tham, C.C. Ko, A probabilistic priority scheduling discipline for multi-service networks, Computer Communications 25 (13) (2002) 1243–1254.
[23] C.K. Tham, Q. Yao, Y. Jiang, A multi-class probabilistic priority scheduling discipline for differentiated services networks, Computer Communications 25 (17) (2002) 1487–1496.
[24] O. Salami, H.A. Chan, Iterative probabilistic scheduling of IP traffic, in: The 3rd IEEE Consumer Communications and Networking Conference, USA, January 8–10, 2006, pp. 1326–1327.
[25] O. Salami, H.A. Chan, M.E. Dlodlo, Probabilistic scheduling of IP traffic, in: The South Africa Telecommunication Networks and Applications Conference, South Africa, September 3–5, 2006.
[26] J. Zhou, C. Beard, Adaptive probabilistic scheduling for a cellular emergency network, in: The 26th IEEE International Performance Computing and Communications Conference, USA, April 11–13, 2007, pp. 612–616.
[27] I. Rao, E.N. Huh, A probabilistic and adaptive scheduling algorithm using system-generated predictions for inter-grid resource sharing, The Journal of Supercomputing 45 (2) (2008) 185–204.
[28] K.S. Trivedi, Probability and Statistics with Reliability, Queuing and Computer Science Applications, 2nd ed., John Wiley and Sons, 2001.
[29] R. Horst, P.M. Pardalos, N.V. Thoai, Introduction to Global Optimization, 1st ed., in: Nonconvex Optimization and its Application, Springer-Verlag, 2002.
[30] Lingo section [Online]. http://www.lindo.com/.
[31] M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, R.F. Freund, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, Journal of Parallel and Distributed Computing 59 (2) (1999) 107–131.
[32] F. Xhafa, L. Barolli, A. Durresi, Immediate mode scheduling of independent jobs in computational grids, in: The 21st International Conference on Advanced Networking and Applications, Canada, May 21–23, 2007, pp. 970–977.
[33] T.D. Braun, 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 61 (6) (2001) 810–837.
[34] G. Khanna, U. Catalyurek, T. Kurc, P. Sadayappan, J. Saltz, A data locality aware online scheduling approach for I/O-intensive jobs with file sharing, in: E. Frachtenberg, et al. (Eds.), Job Scheduling Strategies for Parallel Processing, in: Lecture Notes in Computer Science (LNCS), vol. 4376, Springer, 2006, pp. 141–160.
[35] R. Sahu, A.K. Chaturvedi, Many-objective comparison of twelve grid scheduling heuristics, International Journal of Computer Applications 13 (6) (2011) 9–17.
[36] The Grid Workloads Archive, Workloads/ Gwa-t-4 [Online]. http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Workloads.Gwa-t-4.

Reza Entezari-Maleki is currently a Ph.D. student in computer engineering (software discipline) at the Department of Computer Engineering in Sharif University of Technology in Tehran, Iran. He received his B.S. and M.S. degrees in computer engineering (software discipline) from Iran University of Science and Technology (IUST) in Tehran, Iran in 2007 and 2009, respectively. He is also a member of Iranian National Elite Foundation. His main research interests include grid computing, performance evaluation, performability and dependability modeling, and task scheduling algorithms.

Ali Movaghar received the B.S. degree in electrical engineering from the University of Tehran in 1977 and the M.S. and Ph.D. degrees in computer, information, and control engineering from the University of Michigan, Ann Arbor, in 1979 and 1985, respectively. He is currently a professor in the Department of Computer Engineering at Sharif University of Technology in Tehran, Iran, where he joined first as an assistant professor in 1993. He visited INRIA in France in 1984, worked at AT&T Laboratories from 1985 to 1986, and taught at the University of Michigan from 1987 to 1989. His main areas of interest include performance and dependability modeling, formal verification, wireless and mobile networks, and distributed real-time systems. He is also a senior member of the IEEE and a senior member of the ACM.