International Journal of Engineering and Technology Innovation, vol. 16, no. 4, 2026, pp. 551-567

A Mechanism Design Approach for Dynamic Task Online Scheduling

Xiao-Peng Lin[1], Hua-Ching Chen[1], Hsuan-Ming Feng[2,*]

[1]School of Information Engineering, Xiamen Ocean Vocational College, Xiamen, Fujian 361000, China

[2]Department of Computer Science and Information Engineering, National Quemoy University, Taiwan, R.O.C.

Received 28 August 2026; revised 30 September 2026; accepted 05 October 2026

DOI: https://doi.org/10.46604/ijeti.2026.16741

Abstract

In game-theoretic grid computing, selfish users may misreport task information to obtain individual advantages, leading to prolonged scheduling negotiations and degraded system performance. This study aims to develop the resource allocation mechanism for dynamic tasks (RAMDT), an online scheduling mechanism based on the direct revelation principle. RAMDT incentivizes users to truthfully disclose their private task attributes—value, length, arrival time, and deadline—at the initial bidding stage, thereby accelerating equilibrium convergence, reducing bidding rounds, and minimizing resource waste. Theoretical analysis demonstrates that RAMDT satisfies individual rationality and incentive compatibility, ensuring truthful reporting is a weakly dominant strategy for rational users. Simulation results show that compared with random-bidding protocols, RAMDT reduces the number of rounds for resource equilibrium from exponential to linear in the number of users, and achieves a higher task completion rate than the first-come, first-served (FCFS) scheme. Overall, RAMDT aligns self-interested user behavior with system-wide scheduling efficiency.

Keywords: game theory, grid computing, mechanism design, online task scheduling

1. Introduction

Economic-based grid resource scheduling systems [1–3] rely on user-submitted task information, such as execution budgets and completion deadlines [4–6], to make resource allocation decisions. However, grid users are autonomous and self-interested: under many existing economic scheduling mechanisms [7–9], they may misreport task information to gain individual advantages. Typical examples include reporting artificially tight deadlines to signal greater task urgency and obtain scheduling priority, or starting with low bids and progressively raising them in order to discover favorable prices. The presence of such false information prolongs the strategic bargaining process, increases resource consumption without necessarily improving allocative efficiency, and can lead to erroneous allocation decisions that degrade system-wide performance.

Ex-post approaches such as credibility evaluation [10] can penalize dishonest behavior, but only when objective evidence is available—for instance, verifying actual task sizes or confirming timely payments. However, certain task attributes, such as the completion deadline, are inherently subjective and cannot be independently verified either during or after execution. This does not mean that such attributes lie beyond the reach of mechanism design. Rather than verifying them ex post, one can embed cost structures into the allocation and payment rules ex ante so that misreporting becomes disadvantageous to the user: a user who declares an artificially tight deadline risks having the task cancelled during execution while still bearing the execution cost, yielding negative utility. Under such a design, truthful reporting can emerge as a weakly dominant strategy, aligning individual incentives with system-wide objectives, thereby improving resource utilization and scheduling efficiency.

* Corresponding author. E-mail address: hmfenghmfeng@gmail.com

Economic approaches to grid scheduling have long employed market mechanisms such as commodity markets, auctions, and bargaining protocols [1–3, 11, 12], yet they typically assume that users truthfully report their task requirements. To counter self-interested behavior, game-theoretic and mechanism-design methods have been applied to grid, cloud, and edge resource allocation [10, 13–15]. However, auction-based mechanisms either rely on iterative bidding with high communication overhead [16] or, when single-round [17], leave the incentive issues of multi-attribute private information unresolved. In parallel, online scheduling studies have addressed joint job scheduling and data allocation [7], interactivity-aware scheduling [18], and online mechanism design for demand response [8]. Nevertheless, few existing studies appear to jointly address the incentive problems of multiple private attributes—value, length, arrival time, and deadline—within a single online mechanism, which motivates the mechanism proposed in this paper.

In this paper, a dynamic task scheduling setting is considered in which tasks arrive stochastically over time, are independent of one another (no precedence constraints), and are executed non-preemptively on computational resources that differ only in processing capacity. Within this setting, the resource allocation mechanism for dynamic tasks (RAMDT) is proposed as an online scheduling mechanism grounded in the direct revelation principle of mechanism design theory [19], thereby restricting attention to incentive-compatible direct mechanisms. Each task is characterized by a private type comprising four attributes: arrival time, execution deadline, task length, and willingness-to-pay (which serves as the weight of the task in the system objective). Under RAMDT, a system broker allocates arriving tasks to resource queues based on a value-density criterion and charges each user a compensation cost that internalizes the utility loss inflicted on downstream tasks. Formal analysis establishes that this compensation-based design satisfies both individual rationality and incentive compatibility, and that truthful reporting of all four task attributes, including the unverifiable deadline, is a weakly dominant-strategy equilibrium. Because users no longer benefit from strategic manipulation, the multi-round bidding characteristic of conventional auction-based protocols is eliminated: users submit their true types once and receive allocation decisions in a single round of interaction.

The remainder of this paper is organized as follows. Section 2 formulates the dynamic task scheduling model and defines the user utility function. Section 3 presents the RAMDT mechanism and proves its theoretical properties. Section 4 describes the algorithm implementation and reports the simulation results. Section 5 concludes the paper and discusses future directions.

2. Dynamic Task Scheduling Model

2.1 Model formulation

As grid computing serves a large population of users, tasks from multiple users may arrive simultaneously at any given moment, and tasks arrive at discrete time instants. In an economic grid scheduling system, resource allocation must satisfy user demands while also considering the overall efficiency of resource scheduling. For instance, the system may insert higher-priority tasks while meeting the QoS requirements of other users, or compensate affected users appropriately when impacts occur, ensuring that their utility does not decrease. Accordingly, the mechanism design problem for grid resources is modeled as a dynamic task scheduling problem. Assume that user tasks are distributed discretely over time. Each task i is represented by its type:

$$\theta_i = (a_i, d_i, l_i, w_i) \in T^+ \times T^+ \times R_0^+ \times R_0^+ \tag{1}$$

where $a_i(a_i > 0)$ denotes task arrival time, and $d_i(d_i > a_i + \frac{l_i}{p_j})$ represents the execution deadline, before which the task must be completed. The constraint ensures that the task is feasible on at least the fastest available resource (otherwise it will be rejected). P_j is the processing capacity of resource j, $l_i(l_i > 0)$ is the task size normalized by standard processing capacity, and $w_i(w_i > 0)$ is the willingness-to-pay, representing the user's valuation of the task.

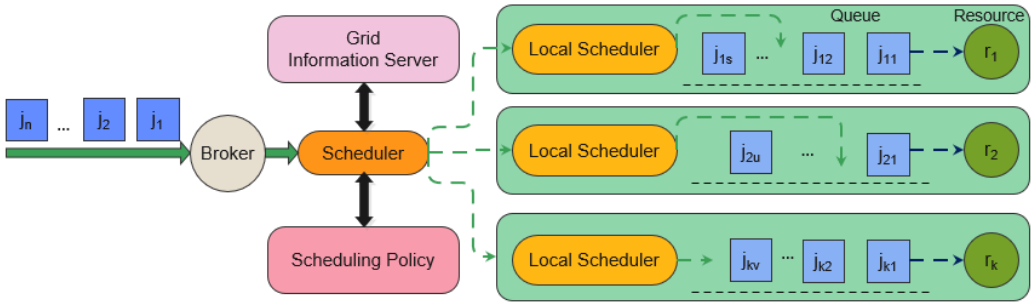


Fig. 1 Dynamic task scheduling model

The type space for user tasks is denoted by $\Theta = \Theta_a \times \Theta_d \times \Theta_l \times \Theta_w$. Assume that grid resources differ only in processing capacity and are identical in other properties, meaning that any task can be executed on any resource, with the processing capacity determining the corresponding execution time. Let the grid resources be M = {1, 2, … m}. Task arrival is a stochastic event following a certain distribution. The scheduling process is illustrated in Fig. 1.

If users submit task information truthfully, the resource scheduling system can make faster and better decisions while satisfying user resource demands, thereby improving overall system performance such as resource utilization. However, since grid users are autonomous, rational, and self-interested, aiming to maximize their own utility, they might conceal their true situation and submit false task values, delaying task submission to gain greater benefits during the resource scheduling process. Let the true type of user's task be $\theta_i = (a_i, d_i, l_i, w_i)$, and the type submitted by the user to the system be $\theta_i' = (a_i', d_i', l_i', w_i')$. The grid resource scheduling system makes resource scheduling decisions based on the submitted task information and the current state of resources.

2.2 Utility function

From the system's perspective, effective grid resource scheduling must balance performance optimization with the influence of users' self-interested behaviors. Through carefully designed mechanisms, users can be incentivized to adopt strategies that enhance overall system performance. In other words, rational decision-making by individual users should simultaneously advance their own utility and contribute to systemic efficiency.

In the proposed model, user behavior involves selecting $\theta_i' = (a_i', d_i', l_i', w_i')$ from the type space Θ according to a certain strategy and submitting it to the resource scheduling system. The goal of mechanism design is to encourage users to submit their true task information $\theta_i = (a_i, d_i, l_i, w_i)$, enabling the resource scheduling system to make appropriate resource allocation decisions.

Based on the premise of user rationality and objective reality, the characteristics of the task information $\theta_i' = (a_i', d_i', l_i', w_i')$ submitted by users are analyzed as follows:

(1) Since tasks cannot be submitted before they arrive, $a_i' \geq a_i$.

(2) The system can obtain task length information during execution and can impose penalties to prevent $l_i' < l_i$; therefore, users will submit $l_i' \geq l_i$.

(3) w_i is completely private information that the system cannot verify. The value w_i' submitted by the user may either reflect the true value ($w_i' = w_i$) or conceal it ($w_i' \neq w_i$).

(4) The task completion deadline is also the user's private information, so the truthfulness of d_i' cannot be discerned by the system.

For convenience of analysis, the following parameters for tasks and the task scheduling process are defined:

(5) The notation $i \rightarrow j$ signifies scheduling task i to resource j.

(6) Let $c_i(j)$ be the completion time of task i on resource j.

(7) The coefficient $k = \frac{w}{l}$ represents the value density of task *i*, which reflects its priority. Tasks in the resource queue are sequenced in descending order of their *k* values. When a new task arrives at resource *j*, it is inserted into the queue at the position corresponding to its *k* value. Tasks with the same k value are ordered by their arrival time.

(8) Let $A_j(t)$ represent the set of all tasks in the task queue of resource *j* at time *t*. For any task *i* in this queue, let $H_i(j)$ denote the set of tasks preceding task *i* in the queue, and $L_i(j)$ denote the set of tasks following task *i* in the queue, i.e.:

$$H_i(j) = \left\{ r \in A_j(t) \mid k_r > k_i \right\} \cup \left\{ s \in A_j(t) \mid_{a_s' > a_i'} k_s = k_i \right\} \tag{2}$$

$$L_i(j) = \left\{ r \in A_j(t) \mid k_r < k_i \right\} \cup \left\{ s \in A_j(t) \mid_{a_s' < a_i'} k_s = k_i \right\} \tag{3}$$

(9) Let $e_j(t)$ denote the remaining execution time of the task currently being executed on resource *j* at time *t*.

Based on the submitted task information, the completion time $c_i'(j)$ required for task *i* to execute on resource *j* is:

$$c_j'(j) = (a_i' - a_i) + e_j(a_i') + \sum_{r \in H_i(j)} \frac{l_r'}{p_j} + \frac{l_i'}{p_j} \tag{4}$$

The remaining time (slack) of task *i* after completion, with respect to its deadline, is given by:

$$t_i(j) = d_i - c_i'(j) = d_i - \left[(a_i' - a_i) + e_j(a_i') + \sum_{r \in H_i(j)} \frac{l_r'}{p_j} + \frac{l_i'}{p_j} \right] \tag{5}$$

The user gains benefit from the execution of task *i*; the longer remaining time indicates higher execution efficiency and thus higher user benefit. Let the user's benefit be $v_i(j) = w_i t_i(j)$, i.e.:

$$v_i(j) = w_i t_i(j) = w_i d_i - w_i \left[(a_i' - a_i) + e_j(a_i') + \sum_{r \in H_i(j)} \frac{l_r'}{p_j} + \frac{l_i'}{p_j} \right] \tag{6}$$

Since a newly arrived task *i* is inserted into the resource's task queue based on its *k* value upon being scheduled, the insertion has no impact on the execution of tasks in set $H_i(j)$. However, the execution times of tasks in set $L_i(j)$ will be prolonged due to the insertion. From Eq. (4), the owners of these tasks will incur a loss of benefits. The owner of task *i* pays compensation to the owners of affected tasks in set $L_i(j)$ to cover these losses. The compensation constitutes the execution cost for task *i*, i.e., the compensation cost for task execution is:

$$\pi_i(j) = \frac{l_i}{p_j} \sum_{r \in L_i(j)} w_i' \tag{7}$$

The utility for the owner of task *i* is the difference between benefit and cost. Consequently, when task *i* is executed on resource *j*, the user's utility function $u_i(j)$ can be expressed as:

$$u_i(j) = v_i(j) - \pi_i(j) \tag{8}$$

i.e.,

$$u_i(j) = w_i d_i - w_i \left[(a_i' - a_i) + e_j(a_i') + \sum_{r \in H_i(j)} \frac{l_r'}{p_j} + \frac{l_i'}{p_j} \right] - \frac{l_i'}{p_j} \sum_{r \in L_i(j)} w_r' \tag{9}$$

The goal of task *i*'s owner is to maximize utility, i.e., given that the task can be executed, select an appropriate type θ_i' from the type space Θ_i such that $u_i(j)$ is maximized.

2.3 Resource allocation mechanism for dynamic tasks

In this model, the user's strategy is to select a specific task type θ_i' from the type space Θ_i and submit it to the resource scheduling system to maximize utility. Here, the user's strategy space is the task type space. A direct mechanism is adopted to implement RAMDT, which is designed to induce truthful reporting of private task information.

The RAMDT mechanism can be applied to online resource scheduling for dynamic tasks. Its execution process is as follows:

(1) Based on the task's utility function $u_i(j)$ and the goal of utility maximization, the user selects an appropriate type $\theta_i^{''} \in \text{argmax}\, u_i(j)$ from the type space Θ_i and submits the task information θ_i' to the resource scheduling system broker;

(2) The broker of the resource scheduling system, based on the information in the user's type θ_i', calculates the task's execution time $c_i'(j)$ and execution cost $\pi_i'(j)$ on various resources, constructs the set of available resources M_J, and returns M_J and $c_i'(j)$ to the user;

(3) To maximize the utility function, the user decides whether to submit the task and selects the resource for execution;

(4) The resource scheduling system schedules the task to the task queue of the user-selected resource where it waits for execution, and charges the user the execution cost;

(5) The maximum completion task is limited to the volume of l_i'. For any remaining portion, it is scheduled to the end of the queue and incurs an additional charge.

Based on the RAMDT execution process, the following cases are considered. If the user does not submit the task, the task execution benefit is zero, the execution cost is zero, and thus the total utility is zero. If the user completes the task by the deadline, the task's benefit is given by Eq. (6). If the submitted task is not completed within the deadline, the user's execution benefit is zero, but they still incur the cost. According to Eq. (9), the user's utility in this case is $u_i(j) = -\frac{l_i'}{p_j}\sum_{r\in L_i(j)} w_r'$. Therefore, the user's utility is:

$$u_i(j)=\begin{cases} w_i d_i - w_i[(a_i^{'}-a_i)+e_j(a_i^{''})+\sum_{r\in H_i(j)}\frac{l_r^{'}}{p_j}+\frac{l_i^{'}}{p_j}]-\frac{l_i^{'}}{p_j}\sum_{r\in L_i(j)} w_r^{'}, & \textit{task is simmitted and completed} \\ -\frac{l_i^{'}}{p_j}\sum_{r\in L_i(j)} w_r^{'}, & \textit{the task is simmitted and not completed} \\ 0, & \textit{the task is not simmitted} \end{cases} \tag{10}$$

2.4 Budget balance

In RAMDT, monetary transfers occur only between users, with the broker acting as a clearing intermediary. When a task is inserted into a resource queue ahead of existing tasks, the inserting user is charged a compensation cost equal to the externality imposed on the delayed downstream tasks, and this amount is disbursed to the affected users as compensation. No payment is created or retained by the broker.

Proposition 1 (Budget Balance). Under RAMDT, the total compensation collected from inserting users equals the total compensation paid to delayed users; the broker runs neither a surplus nor a deficit.

Proof. By construction, the compensation charge levied on an inserting task equals the sum of the externalities imposed on all downstream tasks it delays, and each delayed task receives exactly its imposed externality as compensation. Summing over all insertions, total receipts equal total payments, so the mechanism is ex-post budget balanced.

3. Mechanism Constraints of the RAMDT Algorithm

To ensure optimal resource allocation decisions that improve system performance while meeting users' quality-of-service requirements, it is essential for the resource scheduling system to receive accurate task information from users. In line with principles of mechanism design, the proposed scheduling mechanism must satisfy both incentive compatibility and individual rationality. The following section examines whether and under what conditions the RAMDT mechanism adheres to these constraints.

3.1 Individual rationality constraint

Theorem 1 The RAMDT resource allocation mechanism satisfies the individual rationality constraint.

Proof: $\forall\theta_i \in \Theta,\ a_i,\ d_i,\ l_i,\ w_i$ are all bounded.

(1) If user task i is executed on resource j, then $c_i'(j)$ and $\pi_i'(j)$ are bounded, thus $v_i(j) = -w_i c_i'(j)$, and user's utility $u_i(j) = v_i(j) - \pi_i(j)$ is also bounded.

(2) If the user task cannot be executed, then $c_i'(j) \to \infty,\ v_i'(j) = -w_i c_i'(j) \to -\infty$, thus $u_i'(j) = v_i'(j) - \pi_i'(j) \to -\infty$. The utility from executing a task is greater than that from not executing it, i.e., $u_i(j) > u_i'(j)$. Through mechanism design, rational users are incentivized to participate in task scheduling under this mechanism.

3.2 Incentive compatibility constraint

Theorem 2 The RAMDT resource allocation mechanism satisfies the incentive compatibility constraint.

Theorem 2 is proven by considering the reported task attributes $a_i',\ l_i',\ w_i'$ and d_i' separately.

Lemma 1 For the RAMDT resource allocation mechanism, it is a dominant strategy for a rational user i to submit the task's true value w_i, i.e., $w_i' = w_i$.

Assume at time a_i', user i newly submits a task $\theta_i' = (a_i', d_i', l_i', w_i')$. At that moment, the task queue of resource j contains s tasks, arranged in order of k value. Suppose the k value calculated based on the task i's true value lies between tasks j_u and j_v in the queue, i.e.:

$$\frac{w_1'}{l_1'} > \cdots > \frac{w_u'}{l_u'} > \frac{w_i}{l_i'} > \frac{w_v'}{l_v'} > \cdots > \frac{w_s'}{l_s'} \tag{11}$$

(1) The user's submitted value w_i' satisfies $\frac{w_i'}{l_i'} > \frac{w_v'}{l_v'} > \cdots > \frac{w_s'}{l_s'}$.

As shown in Fig. 2(a), according to the utility function in Eq. (9), if task i is scheduled to resource j and added at the end position s+1, the utility is:

$$u_i^{(s+1)}(j) = w_i d_i - w_i \left[(a_i' - a_i) + e_j(a_i') + \sum_{r=1}^{s} \frac{l_r'}{p_j} + \frac{l_i'}{p_j} \right] \tag{12}$$

If task i at position s+l is swapped with task j_s at position s in the original queue, as shown in Fig. 2(b), the utility becomes:

$$u_i^{(s)}(j) = w_i \left[d_i - \left((a_i' - a_i) + e_j(d_i') + \sum_{r=1}^{s-1} \frac{l_r'}{p_j} + \frac{l_i'}{p_j} \right) \right] - \frac{l_i'}{p_j} w_s' \tag{13}$$

where the compensation for the affected task j_s after insertion is included in the sum over $\frac{l_i'}{p_j} w_s'$.

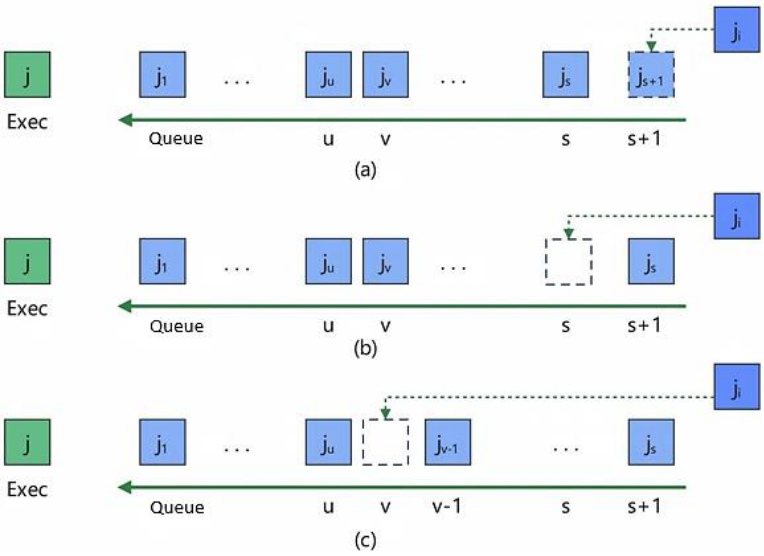


Fig. 2 User ranking in the queue based on submitted value

Let $\Delta u_i^{(1\to m)}(j)$ represent the change in utility when task *i* moves from position *l* to *m* in the queue of resource *j*. Based on the utilities of the task at positions *s* and *s+l* in Eqs. (12) and (13), the change is:

$$\Delta u_i^{(s+1\to s)}(j)=u_i^{(s)}(j)-u_i^{(s+1)}(j)=w_i\frac{l_s^{'}}{p_j}-\frac{l_i^{'}}{p_j}w_s^{'}=\frac{l_i^{'}l_s^{'}}{p_j}\left(\frac{w_i}{l_i^{'}}-\frac{w_s^{'}}{l_s^{'}}\right)>0 \tag{14}$$

As is evident, swapping the positions of tasks j_i and j_s yields a higher utility for user *i* than queuing at the end of the line. Consequently, a rational user, when submitting a task $\theta_i'=(a_i',d_i',l_i',w_i',)$, should strategically act to ensure $\frac{w_i'}{l_i'}>\frac{w_s'}{l_s'}$, that j_i is placed ahead of j_s in the queue.

Similarly, for a task at queue position *s*, when condition $\frac{w_i}{l_i'}>\frac{w_{s-1}'}{l_{s-1}'}$ holds, swapping the task j_i at position *s* with j_{s-1} at position *s*-1 yields the change in utility:

$$\Delta u_i^{(s\to s-1)}(j)=\frac{l_i^{'}l_{s-1}^{'}}{p_j}\left(\frac{w_i}{l_i^{'}}-\frac{w_{s-1}^{'}}{l_{s-1}^{'}}\right)>0 \tag{15}$$

Thus, the rational user submitting the task should choose $\frac{w_i'}{l_i'}>\frac{w_{s-1}'}{l_{s-1}'}$. By analogy, the task satisfying submitted $\frac{w_1'}{l_1'}>\cdots>\frac{w_u'}{l_u'}>\frac{w_i}{l_i'}>\frac{w_v'}{l_v'}>\cdots>\frac{w_s'}{l_s'}$ value should satisfy $\frac{w_i'}{l_i'}>\frac{w_v'}{l_v'}>\cdots>\frac{w_s'}{l_s'}$, to ensure task *i* can be placed before j_v in the queue, as shown in Fig. 2(c). That is:

$$\frac{w_i^{'}}{l_i^{'}}>\frac{w_v^{'}}{l_v^{'}}>\cdots>\frac{w_s^{'}}{l_s^{'}} \tag{16}$$

(1) The user's submitted value w_i' satisfies $\frac{w_1'}{l_1'}>\cdots>\frac{w_u'}{l_u'}>\frac{w_i'}{l_i'}$.

If the submitted value of task *i* satisfies $\frac{w_i'}{l_i'}>\frac{w_u'}{l_u'}$, that is, the task *i* at queue position *u*+l swaps its queue order with task j_u at position u, the additional utility gained by task *i* is:

$$\Delta u_i^{(u+1\to u)}(j)=u_i^{(u)}(j)-u_i^{(u+1)}(j)=w_i\frac{l_u^{'}}{p_j}-\frac{l_i^{'}}{p_j}w_u^{'}=\frac{l_i^{'}l_u^{'}}{p_j}(\frac{w_i}{l_i^{'}}-\frac{w_u^{'}}{l_u^{'}})<0 \tag{17}$$

It is evident that if a user posts a task with an inflated value to gain priority execution, the utility obtained by the user will decrease instead. Therefore, the value posted by the user should be $\frac{w_1'}{l_1'} > \cdots > \frac{w_u'}{l_u'} > \frac{w_i'}{l_i'}$.

From the above proofs, the value submitted by the user for the task should satisfy:

$$\frac{w_1^{'}}{l_1^{'}} > \cdots > \frac{w_u^{'}}{l_u^{'}} > \frac{w_i^{'}}{l_i^{'}} > \frac{w_v^{'}}{l_v^{'}} > \cdots > \frac{w_s^{'}}{l_s^{'}} \tag{18}$$

From Eqs. (16) and (18), a rational user's submitted task value w_i' should place the task in the same queue position as if submitted with the true value. Since tasks arrive dynamically, to ensure task *i* remains in the position corresponding to its true value even after subsequent task arrivals, submitting the true value aligns with the user's optimization goal.

Besides considering the change in utility for task *i*, the impact on the utilities of other tasks behind the insertion point must be considered. For the aforementioned insertion of task *i* before j_s, the change in utility for task j_s is:

$$\Delta u_s^{(s \to s+1)} = \Delta v_s(j) + \Delta \pi_s(j) = -w_s \Delta c_s(j) + w_s^{'} \frac{l_i^{'}}{p_j} = -w_s \frac{l_i^{'}}{p_j} + w_s^{'} \frac{l_i^{'}}{p_j} = \frac{l_i^{'}}{p_j}(w_s^{'} - w_s) \tag{19}$$

Since the user submits the true value, $\Delta u_s^{(s \to s+1)} = 0$. That is, the utilities of existing tasks remain unchanged after swapping positions with task j_i. Consequently, true value reporting guarantees compensation-neutrality: any task displaced by a higher-priority insertion receives full compensation for the externality imposed on it, ensuring that the utility of the displaced user does not decrease. A full characterization of allocative Pareto efficiency across all users is left as future work.

In summary, for the types θ'_{-j} submitted by other users, user *i* will submit a type with the true value w_i, i.e., $w_i' = w_i$.

Lemma 2 The RAMDT makes it a dominant strategy for a rational user *i* to submit the true task length, i.e., $l_i' = l_i$.

Proof: Since the resource can obtain task length information during execution, penalties can be imposed for tasks with $l_i' < l_i$, such as rescheduling the remaining part of the task $l_i - l_i'$ to the end of the queue after execution starts l_i', increasing the task's execution time $c_i(j)$ and reducing the user's utility. Therefore, rational users should submit $l_i' \geq l_i$ when submitting the task.

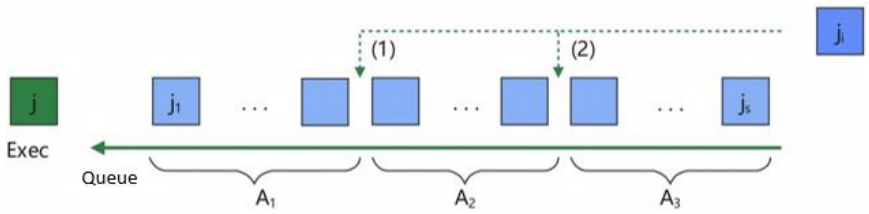


Fig. 3 Queue ordering based on the submitted task length

If the task is scheduled according to $\frac{w_i'}{l_i}$, the task will be at position μ in the queue, with utility $u_i(j)$; if scheduled according to $\frac{w_i'}{l_i'}$, it will be at position μ, with utility $u_i'(j)$. Based on the two insertion points, the original queue tasks are divided into three sets A1, A2, and A3, i.e., $A_1 = \left\{ j_r \in A \middle| \frac{w_r'}{l_r'} > \frac{w_i'}{l_i} \right\}$, $A_2 = \left\{ j_r \in A \middle| \frac{w_i'}{l_i} > \frac{w_r'}{l_r'} > \frac{w_i'}{l_i'} \right\}$, $A_3 = \left\{ j_r \in A \middle| \frac{w_i'}{l_i'} > \frac{w_r'}{l_r'} \right\}$, as shown in Fig. 3.

According to Eq. (9):

$$u_i(j) - u_i^{'}(j) = \frac{w_i}{p_j}(l_i^{'} - l_i) + \frac{\sum_{r \in A_3} w_r^{'}}{p_j}(l_i^{'} - l_i) + \sum_{r \in A_2} \left(\frac{l_r^{'} l_i}{p_j} \left(\frac{w_i}{l_i} - \frac{w_r^{'}}{l_r^{'}} \right) \right) \tag{20}$$

According to Lemma 1, all users submit the true value, i.e., $w_i' = w_i$, and thus:

$$u_i(j) - u_i'(j) = \frac{w_i}{p_j}(l_i' - l_i) + \frac{\sum_{r\in A_3} w_r'}{p_j}(l_i' - l_i) + \sum_{r\in A_2}\left(\frac{l_r' l_i}{p_j}\left(\frac{w_i'}{l_i} - \frac{w_r'}{l_r'}\right)\right) > 0 \tag{21}$$

It follows that the utility $u_i(j)$ obtained by a user when submitting a task with the true length l_i is greater than $u_i'(j)$ that is obtained when using a false length l_i'. Consequently, a rational user will submit tasks using the true length l_i.

Lemma 3 RAMDT makes it a dominant strategy for a rational user *i* to submit tasks at their actual arrival time, i.e., $a_i' = a_i$.

Proof: Since a task can only be submitted after it arrives, $a_i' \geq a_i$. Suppose two identical tasks j_i and j_s arrive simultaneously at time a_i, $a_i = a_s$, $w_i = w_s$, $w_i' = w_s'$, $l_i' = l_s'$, but task j_i is submitted immediately at time d_i, while task j_s is submitted at time d_i' ($d_i' > d_i$).

Let A_N be the set of other tasks submitted in the time interval $[d_i, d_i']$. Among A_N, let A_H and A_L denote tasks scheduled into the queue before j_i and after j_i. According to Eq. (9), the utilities of tasks j_i and j_s are given by:

$$u_i(j) = v_i(j) - \pi_i(j) = w_i d_i - w_i\left[e_j(a_i) + \sum_{r\in H_i(j)}\frac{l_r'}{p_j} + \frac{l_i'}{p_j}\right] - \frac{l_i'}{p_j}\sum_{r\in L_i(j)} w_r' \tag{22}$$

$$u_s'(j) = v_s'(j) - \pi_i'(j) = w_i d_i - w_i\left[(a_s' - a_i) + e_j(a_s') + \sum_{r\in H_s'(j)}\frac{l_r'}{p_j} + \frac{l_s'}{p_j}\right] - \frac{l_s'}{p_j}\sum_{r\in L_s'(j)} w_r' \tag{23}$$

(1) If task j_s is submitted while j_i remains in the queue.

Under the RAMDT mechanism, task j_s will be positioned after j_i in the queue. Given that $L_i(j) \subseteq L_s'(j)$, then $\pi_i(j) \leq \pi_s'(j)$. Since j_i remains in the task queue, the resources remain occupied during the time interval $[d_i, d_i']$. Therefore:

$$a_s' - a_i = e_j(a_i) + \sum_{r\in\left(H_s(j)\cup A_H(j) - H_s'(j)\right)}\frac{l_r'}{p_j} - e_j(a_s') \tag{24}$$

Then:

$$a_s' - a_i = e_j(a_i) + \sum_{r\in H_i(j)}\frac{l_r'}{p_j} + \sum_{r\in A_H(j)}\frac{l_r'}{p_j} - \sum_{r\in H_s'(j)}\frac{l_r'}{p_j}(a_s') - e_j(a_s') \tag{25}$$

$$e_j(a_i) + \sum_{r\in H_i(j)}\frac{l_r'}{p_j} + \sum_{r\in A_H(j)}\frac{l_r'}{p_j} = (a_s' - a_i) + \sum_{r\in H_s'(j)}\frac{l_r'}{p_j}(a_s') - e_j(a_s') \tag{26}$$

so:

$$-w_i\left[e_j(a_i) + \sum_{r\in H_i(j)}\frac{l_r'}{p_j} + \frac{l_i'}{p_j}\right] \geq -w_i\left[(a_s' - a_i) + e_j(a_s') + \sum_{r\in H_s'(j)}\frac{l_r'}{p_j} + \frac{l_s'}{p_j}\right] \tag{27}$$

Then $v_i(j) \geq v_s'(j)$, because $u_i(j) = v_i(j) - \pi_i(j)$ and $u_s'(j) = v_s'(j) - \pi_i'(j)$, therefore $u_i(j) \geq u_s'(j)$.

When $A_H = \emptyset$, then $u_i(j) = u_s'(j)$, else $u_i(j) > u_s'(j)$.

(1) If task j_s is submitted after j_i has left the queue.

Given that task j_i is not in the queue, it must have been completed or removed from execution at that time. Then:

$$c_i(j) \le (a_i^{'} - a_i) + e_j(a_i^{'}) \tag{28}$$

which yields $u_i(j) = u_s'(j)$.

Therefore, submitting the task at time $\mathrm{d_i}$ is a dominant strategy relative to submission at time $d_s'(d_s' > d_i)$. A rational user therefore has an incentive to submit the task at time $\mathrm{d_i}$. Lemma 3 is proven.

Lemma 4 RAMDT makes it a weakly dominant strategy for a rational user *i* to submit the task type based on its actual completion deadline, i.e., $d_i' = d_i$.

Proof: The RAMDT schedules tasks into the resource queue using $k_i' = \frac{w_i'}{l_i'}$ as the reference metric, without considering the actual completion deadline. However, task execution takes the actual deadline into account. If the task has not been completed by the time $t = d_i'$, its execution is canceled, while the user is still charged. From Eq. (10), the task utility in this case is negative, $u_i(j) = -\frac{l_i'}{p_j}\sum_{r \in L_i(j)} w_r'$.

If the user adopts strategy $d_i' < d_i$, this strategy is weakly dominated by truthful submission: it yields identical utility when the task is completed before i^d and strictly lower utility when completion falls between i^d and d_i. This demonstrates that strategy $d_i' < d_i$ constitutes a dominated strategy for the user. Similarly, submitting $d_i' > d_i$ is also weakly dominated: it yields identical utility when the task completes before d_i and strictly lower utility when it completes between d_i and i^d. Therefore, the user's weakly dominant strategy is to submit $d_i' = d_i$. Lemma 4 is proven.

Combining Lemmas 1-4, it follows that truthful reporting of the task information, together with prompt submission upon arrival, is a weakly dominant strategy for the user. Therefore, rational users will submit $\theta_i = (a_i, d_i, l_{i,} w_i)$. Theorem 2 is fully proven. More precisely, truthful reporting is weakly dominant overall: for the deadline attribute (Lemma 4), any misreport is only weakly dominated by truthful reporting, whereas for the value, length, and arrival-time attributes (Lemmas 1-3), truthful reporting is strictly dominant.

The user utility can be represented by a tree structure as shown in Fig. 4. When the user submits the task with true information, the utility is $u_i(j)$ and $0 \le u_i(j)$; when submitting under other conditions , the utilities are $u_i'(j)_1$, $u_i'(j)_2$, $u_i'(j)_3$ and $u_i'(j)_1 \le 0 \le u_i(j)$, $u_i'(j)_2 \le 0 \le u_i(j)$ and $u_i'(j)_3 \le 0 \le u_i(j)$; if the task is not submitted, the utility is 0. Therefore, the utility-maximizing strategy is to submit the task type truthfully. If the task can be completed within the deadline, the user follows strategy 1 in Fig. 4; if it exceeds the deadline, the user follows strategy 2 in Fig. 4 and abandons task execution.

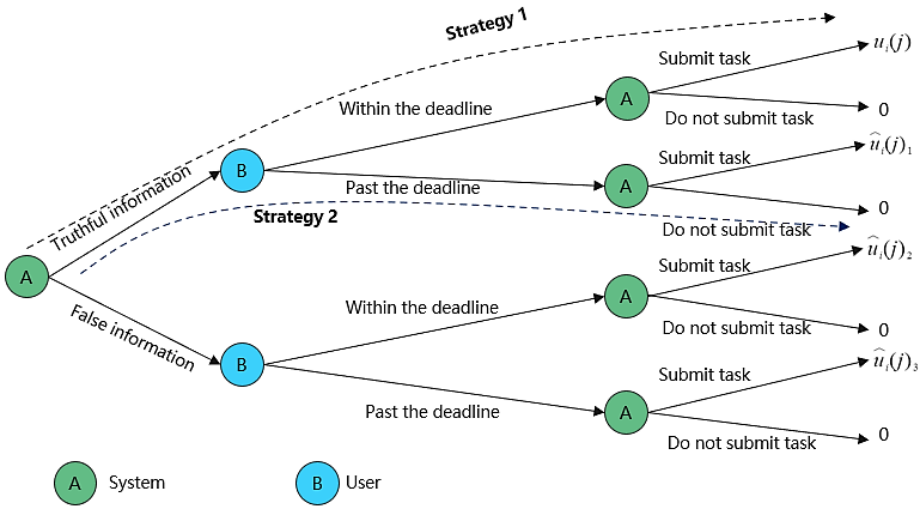


Fig. 4 Tree representation of user utility

4. Simulation and Analysis

4.1 Algorithm implementation of the mechanism

From the above analysis, the resource allocation mechanism for dynamic tasks (RAMDT), implemented as a direct mechanism, can incentivize users to submit true task information. The implementation of the RAMDT consists of two parts: the user strategy for dynamic task (USDT) algorithm on the user side and the broker resource allocation algorithm (BRAA) on the resource scheduling system broker side.

Under the RAMDT resource scheduling mechanism, rational users perform task type submission, decide whether to submit the task, and select the resource, as shown in Fig. 5. The USDT algorithm is implemented as shown in Algorithm 1.

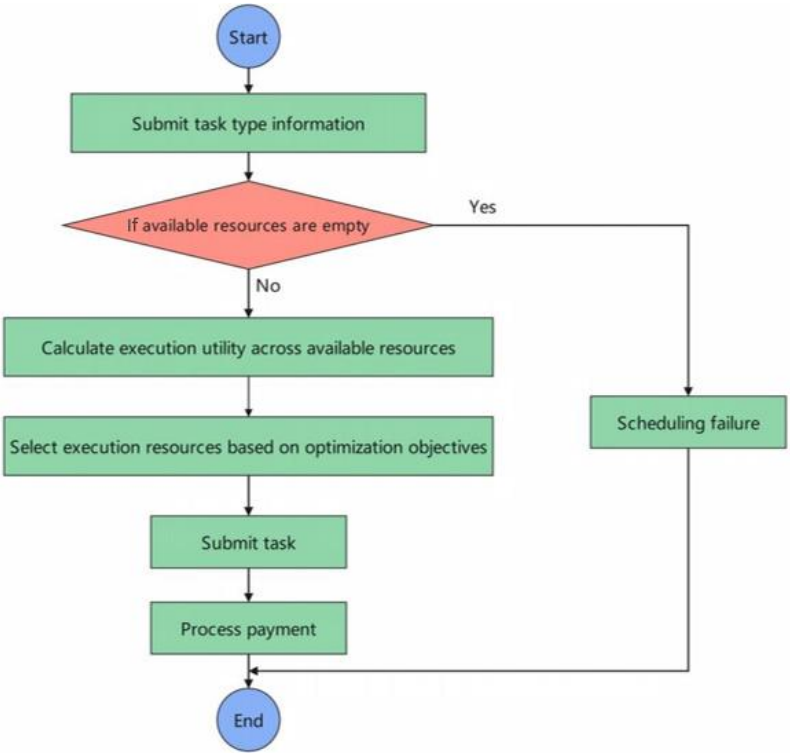


Fig. 5 User task strategy selection process

Algorithm 1 USDT Algorithm (User Strategy for Dynamic Task)

1. Submit task type $\theta_i' = (a_i', d_i', l_i', w_i',)$
2. Receive from resource scheduling system: available resource set M_J, $c_i(j)$, $\pi_i(j)$
3. If set M_J is empty **Then**
4. Return False
5. Else
6. Calculate task utility $u_i(j)$ on each available resource using Eq. (10)
7. Select resource j such that $u_i(j)$ is maximized
8. Submit resource selection result to resource scheduling system
9. Pay cost $c_i(j)$ to the system
10. End if
11. Return True

The resource scheduling system broker, based on the RAMDT mechanism, user task information, and the user's selected resource, schedules the task to the corresponding resource queue and charges the execution cost. The task scheduling process of the resource scheduling system is illustrated in Fig. 6. The BRAA is implemented as shown in Algorithm 2.

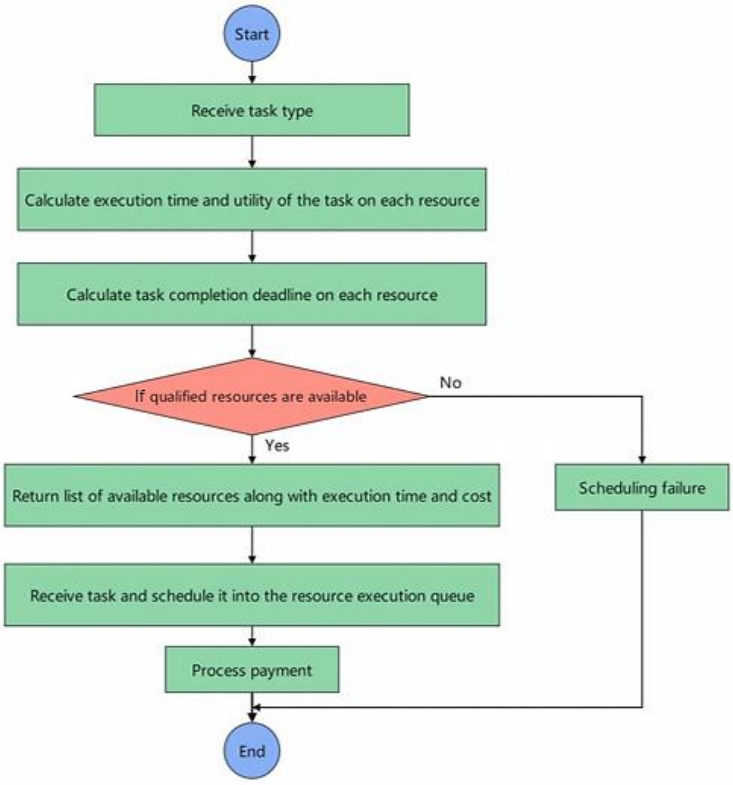


Fig. 6 The task scheduling process of the resource scheduling system

Algorithm 2 BRAA Algorithm (Broker Resource Allocation Algorithm)

1. Receive the user's task type
2. Calculate the task's completion time using Eq. (4)
3. Calculate the task's execution cost using Eq. (7)
4. If set M_J is empty **Then**
5. Return False
6. Else
7. Form the available resource set M_J
8. Obtain the user's resource selection.
9. Receive task and schedule to selected resource.
10. Charge the user the execution cost
11. End if
12. Return True

Computational complexity of BRAA. For a resource queue of length n, determining the insertion position by criticality (value density) takes O(log n) time via binary search, while shifting the queue costs O(n); computing the compensation charge requires scanning at most n tasks behind the insertion point, i.e., O(n) per arriving task. The broker therefore incurs O(n) time and O(n) space per task in the worst case. This per-task cost can be reduced to O(log n) per insertion by using a heap- or balanced-tree-based queue structure. Since the design is centralized around a single broker, its scalability bottleneck lies in the broker's per-task overhead.

The Remark (Broker overhead in practice). The per-task broker cost is dominated by a single linear scan of at most n queued tasks with O(1) arithmetic per entry. Even for a queue of n = 10,000 tasks, this corresponds to approximately 10^4-10^5 elementary operations, which may require only microseconds on commodity hardware. By contrast, task inter-arrival times in the target grid workloads are on the order of milliseconds to seconds (a mean inter-arrival time of $\frac{1}{\lambda} \approx 143$ ms at $\lambda = 7$ in the simulations). The broker's sustainable throughput thus substantially exceeds the task arrival rate, and no significant queueing

at the broker is expected under the studied conditions. Systematic throughput benchmarking under extreme arrival rates, together with the distributed multi-broker design discussed in the Conclusions, is left as future work.

4.2 Simulation and analysis

Simulation setup. The simulated grid comprises 100 users, 10 resources, and 1000 tasks per configuration. Task arrivals follow a Poisson process with rate λ = 7; task lengths are normally distributed with mean 1.0 and standard deviation 0.2; task values and deadlines are drawn from Uniform(1, 10) and Uniform(5, 50), respectively. Each configuration is repeated over K = 30 independent trials, and all results are reported as the mean values with 95% confidence intervals.

Positioning relative to VCG-style and auction-based mechanisms. A direct numerical comparison with the classical VCG mechanism is not directly appropriate in this setting: VCG is an offline mechanism that requires all agent types to be reported before allocation and does not address unverifiable private information, whereas RAMDT makes irrevocable online decisions upon each task arrival and specifically incentivizes truthful reporting of an unverifiable deadline. Indeed, the compensation charge of RAMDT is itself a VCG-style externality payment adapted to the online setting; RAMDT can therefore be viewed as an online analogue of VCG rather than a competitor. Table 1 contrasts RAMDT with the classical VCG mechanism and representative auction-based mechanisms [10, 13-15] at the feature level. Given these structural differences, the numerical experiments below compare RAMDT with the online baselines of FCFS and random bidding.

The resource allocation mechanism is evaluated in this model, focusing on the selection of a_i', d_i', l_i', w_i' in the user task type space to illustrate the utility changes. Task arrival times follow a Poisson distribution, and task lengths follow a normal distribution. The sequence of task scheduling is illustrated in Fig. 7.

Table 1 Feature-level comparison of RAMDT with VCG-style and auction-based mechanisms

Feature	RAMDT	Classical VCG	Auction-based mechanisms
Decision timing	Online, irrevocable upon arrival	Offline; all types known in advance	Offline or multi-round iterative
Reporting protocol	Single-round direct revelation of four attributes	Single-shot sealed bid	Multi-round bidding
Unverifiable deadline	Incentivized via ex-ante externality charge	Not addressed	Not addressed
Payment rule	Compensation = externality on downstream tasks (VCG-style, online analogue)	Clarke pivot externality payment	Price-based payments
Budget balance	Exact balance (Proposition 1)	Not guaranteed in general	Depends on design
Communication overhead	O(1) reports per task	O(1) reports, but offline	Multiple bidding rounds

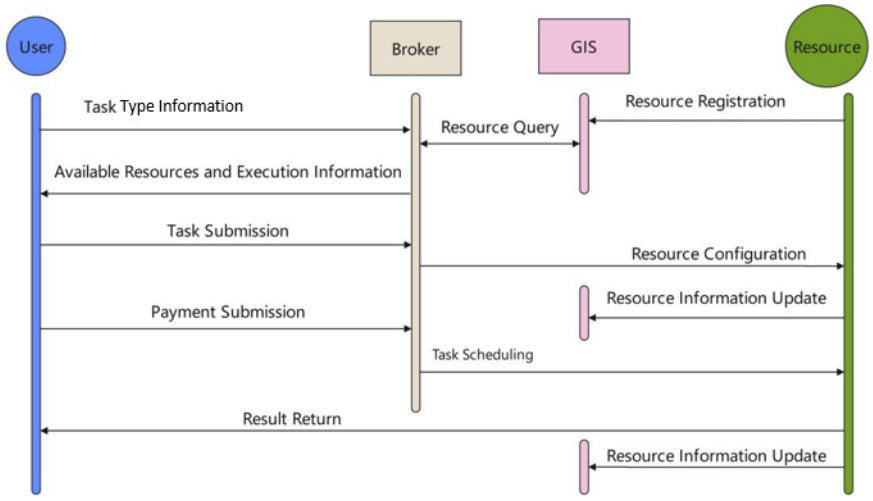
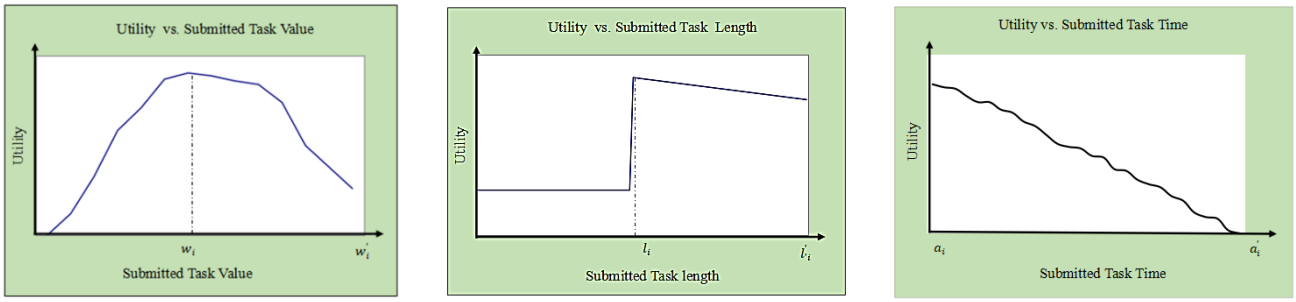


Fig. 7 Dynamic task resource configuration sequence

The simulation environment includes users, the system broker, the grid information server (GIS), and resources. Resources register their relevant information with GIS when joining the grid system. Task generation is modeled as a random event. When a task is generated, the user submits the task type information to the system broker. The Broker uses the BRAA algorithm to select and allocate resources for the task and returns the execution time and cost for the available resources to the user. If the user submits the task, actual resource allocation and scheduling are performed, and the corresponding cost is charged. Task execution results are returned directly from the resource to the user. After a resource is allocated or released, the resource entity updates its information in the GIS.

The user chooses a task type $\theta_i' = (a_i', d_i', l_i', w_i')$ from type space Θ_i, and the utility variation with the submitted value w_i' is shown in Fig. 8(a).



(a) Utility vs. submitted task value (b) Utility vs. submitted task length (c) Utility vs. submitted task time

Fig. 8 Relationship between utility and submitted task value, length, and time

Simulation results show that, with other information unchanged, the user's utility varies with the submitted value w_i'. When $w_i' = w_i$, the utility reaches its maximum, indicating that submitting the true value w_i aligns with the user's interest. For a specific user selecting type $\theta_i' = (a_i', d_i', l_i', w_i')$ from task i's type space Θ_i, the relationship between utility and the submitted task length l_i' is shown in Fig. 8(b). The simulation results show that, with other task attributes held constant, the user's utility changes accordingly with the variation in the submitted task length l_i'. When $l_i' = l_i$, the utility reaches its maximum, indicating that truthful reporting of the task length maximizes the user's utility.

The relationship between the task submission time and user utility is illustrated in Fig. 8(c). Task generation within a certain period is assumed to be a random event following a Poisson distribution. Since a task can only be submitted after it arrives, $t = 0$ represents the task arrival time, and $t > 0$ denotes the delayed submission time.

According to Eq. (9), delaying task submission does not increase the task's benefit, $v_i(j)|_{t=0} \geq v_i(j)|_{t>0}$. Meanwhile, other tasks submitted during the time interval [0, t] increase the execution cost of the original task, $u_i(j)|_{t=0} \geq u_i(j)|_{t>0}$. Therefore, the user's utility gradually decreases with an increase in delay time, as illustrated in Fig. 8(c). When $a_i' + c_j(j) > d_i$, the task cannot be completed within its deadline. Since task execution under this condition yields negative utility, a rational user would not submit the task, resulting in zero utility.

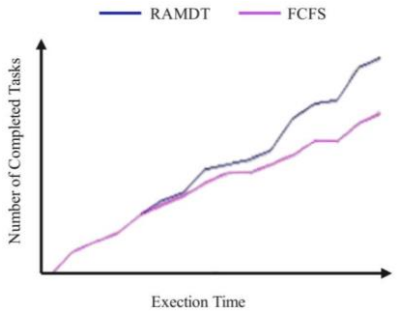


Fig. 9 Comparison between RAMDT and FCFS algorithms

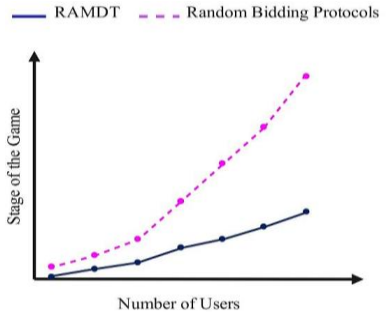


Fig. 10 Comparison between RAMDT and random bidding protocols

In RAMDT, task criticality is determined by the value density k; the mechanism dynamically inserts tasks into the scheduling queue according to their criticality levels while compensating affected users. Provided that the scheduled tasks can be completed before their deadlines, tasks with higher criticality are given priority. In contrast, the FCFS scheduling algorithm processes tasks strictly in order of arrival. Even if a later-arriving task has a short deadline, it must wait in the queue, which may cause it to miss its deadline and remain incomplete.

A comparison between the RAMDT mechanism and the FCFS algorithm is presented in Fig. 9. The results demonstrate that, by prioritizing critical tasks, the RAMDT mechanism achieves a higher number of successfully executed tasks under identical conditions, thereby attaining a higher task completion rate than the FCFS scheduling algorithm.

In the random-bidding baseline, each user submits a bid drawn independently from a uniform distribution in each round. Users who do not win a resource re-bid in the subsequent round, and the process terminates when the allocation remains unchanged between consecutive rounds, which is defined as equilibrium.

For conventional random bidding protocols, the number of rounds required to reach resource equilibrium approximates an exponential relationship with the number of users, as shown in Fig. 10. In contrast, when the RAMDT approach is adopted, the number of bidding rounds required to reach equilibrium exhibits a linear relationship with the number of users, demonstrating that RAMDT can achieve rapid resource matching.

5. Conclusions

The study investigated incentive-compatible online scheduling of dynamic tasks in economic grid systems, where autonomous and self-interested users may misreport private task information to gain scheduling advantages. A dynamic task scheduling model was formulated in which each task is characterized by four private attributes—arrival time, execution deadline, task length, and willingness-to-pay—and the resource allocation mechanism for dynamic tasks (RAMDT), a direct-revelation online mechanism, was implemented through the user-side USDT algorithm and the broker-side BRAA algorithm. Theoretical properties of the mechanism were formally established, and its performance was evaluated through simulation. The main conclusions of this study are summarized as follows:

(1) RAMDT satisfies individual rationality: whether a task is executed or not, a participating user's utility is always non-negative, so rational users are willing to join the mechanism.
(2) RAMDT satisfies incentive compatibility: truthful and prompt reporting of all four private attributes—value, length, arrival time, and deadline—is a weakly dominant strategy. Misreporting, including inflating the task value, understating the task length, delaying submission, or declaring an artificially tight deadline, cannot increase user utility.
(3) By eliminating the incentive for strategic manipulation, RAMDT replaces the multi-round bargaining of conventional auction-based protocols with a single round of truthful type submission, thereby reducing communication overhead and accelerating the convergence of resource equilibrium.
(4) Simulation results show that the number of rounds required to reach resource equilibrium grows linearly, rather than exponentially, with the number of users, and that RAMDT achieves a higher task completion rate than the FCFS algorithm under identical conditions.

This study has several limitations that point to promising directions for future research. First, our model assumes that tasks are independent and non-preemptive; extending the mechanism to tasks with precedence constraints or preemption would broaden its applicability. Second, resources are assumed to differ only in processing capacity; heterogeneous resources with varying cost structures or specialized capabilities remain to be addressed. Third, the simulation environment assumes that task arrival distributions are known; in practice, these distributions may be unknown or time-varying, motivating adaptive or learning-enhanced mechanism designs. Future work will explore these extensions, as well as the integration of RAMDT with

real-world grid testbeds to validate its performance under realistic workloads. As a concrete path, heterogeneous resources can be modeled using a multi-dimensional resource vector (e.g., CPU, memory, and bandwidth), with the compensation charge defined as the sum of the per-dimension externalities; task precedence can be represented by a directed acyclic graph that restricts feasible insertion positions. Both extensions keep the ex-ante externality-charge framework and the proofs of Lemmas 1-3 intact, while Lemma 4 needs to be re-verified under the constrained insertion rule. A systematic sensitivity analysis of the task arrival rate and the number of resources is also left for future work.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] A. J. Jadhav and G. A. Patil, "Optimizing Computational Resources for Improved Accuracy in Grid Computing Environments," International Journal of Advance Computational Engineering and Networking (IJACEN), vol. 12, no. 9, pp. 93–98, 2024.

[2] R. Wolski, J. S. Plank, J. Brevik, and T. Bryan, "Analyzing Market-Based Resource Allocation Strategies for the Computational Grid," International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 258–281, 2001.

[3] R. Buyya, D. Abramson, and S. Venugopal, "The Grid Economy," Proceedings of the IEEE, vol. 93, no. 3, pp. 698–714, 2005.

[4] I. Foster and C. Kesselman, Eds., The Grid 2: Blueprint for a New Computing Infrastructure, 2nd ed., San Francisco, CA, USA: Morgan Kaufmann, 2004.

[5] X. P. Lin, X. C. Li, and D. H. Guo, "Principles and Key Technologies of Grid Computing," Communications Technology, vol. 42, no. 2, pp. 218–220, 2009 (in Chinese).

[6] Z. T. Jiang, K. Li, Y. Wang, M. L. Liu, and H. Q. Li, "A Task Allocation Schema Based on Response Time Optimization in Cloud Computing," Cluster Computing, vol. 27, no. 3, pp. 3893–3910, 2024.

[7] S. Feng, J. Kim, Y. Yang, J. Boudreau, T. Chowdhury, A. Hoisie, et al., "Alternative Mixed Integer Linear Programming Optimization for Joint Job Scheduling and Data Allocation in Grid Computing," Future Generation Computer Systems, vol. 175, article no. 108075, 2026.

[8] S. Chen, L. Jiao, F. Liu, and L. Wang, "EdgeDR: An Online Mechanism Design for Demand Response in Edge Clouds," IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 2, pp. 343–358, 2022.

[9] A. B. MacKenzie and L. A. DaSilva, Game Theory for Wireless Engineers. San Rafael, CA, USA: Morgan & Claypool Publishers, 2006.

[10] Y. K. Kwok, K. Hwang, and S. S. Song, "Selfish Grids: Game-Theoretic Modeling and NAS/PSA Benchmark Evaluation," IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 5, pp. 621–636, 2007.

[11] E. N. Alkhanak, S. P. Lee, R. Rezaei, and R. M. Parizi, "Cost Optimization Approaches for Scientific Workflow Scheduling in Cloud and Grid Computing: A Review, Classifications, and Open Issues," Journal of Systems and Software, vol. 113, pp. 1–26, 2016.

[12] M. B. Qureshi, M. M. Dehnavi, N. Min-Allah, M. S. Qureshi, H. Hussain, I. Rentifis, et al., "Survey on Grid Resource Allocation Mechanisms," Journal of Grid Computing, vol. 12, no. 2, pp. 399–441, 2014.

[13] M. G. Buscemi, U. Montanari, and S. Taneja, "A Game-Theoretic Analysis of Grid Job Scheduling," Journal of Grid Computing, vol. 10, no. 3, pp. 501–509, 2012.

[14] F. Xhafa and J. Kolodziej, "Game-Theoretic, Market and Meta-Heuristics Approaches for Modelling Scheduling and Resource Allocation in Grid Systems," Proceedings of the 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), IEEE, pp. 235–242, 2010.

[15] N. Sharghivand, F. Derakhshan, and N. Siasi, "A Comprehensive Survey on Auction Mechanism Design for Cloud/Edge Resource Management and Pricing," IEEE Access, vol. 9, pp. 126502–126529, 2021.

[16] H. Prasad, A. Anand, S. S. Sudha, S. Choudhury, M. K. Gourisaria, and A. Bandyopadhyay, "Game Theory-Based Auction Approach for Resource Allocation in Grid Computing Using Microgrids," Proceedings of the 2024 Global Conference on Communications and Information Technologies (GCCIT 2024), IEEE, pp. 1–6, 2024.

[17] A. Salehan, H. Deldari, and S. Abrishami, "An Online Valuation-Based Sealed Winner-Bid Auction Game for Resource Allocation and Pricing in Clouds," The Journal of Supercomputing, vol. 73, no. 11, pp. 4868–4905, 2017.

[18] J. Tadrous, A. Eryilmaz, and A. Sabharwal, "Action-Based Scheduling: Leveraging App Interactivity for Scheduler Efficiency," IEEE/ACM Transactions on Networking, vol. 27, no. 1, pp. 112–125, 2019.

[19] T. Börgers, D. Krähmer, and R. Strausz, An Introduction to the Theory of Mechanism Design. Oxford, U.K.: Oxford University Press, 2015.



Copyright© by the authors. Licensee TAETI, Taiwan. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY-NC) license (https://creativecommons.org/licenses/by-nc/4.0/).