Original citation:
Chang, Cheng, He, Ligang, Chaudhary, Nadeem, Fu, Songling, Chen, Hao, Sun, Jianhua, Li, Kenli, Fu, Zhangjie and Xu, Ming-Liang. (2017) Performance analysis and optimization for workflow authorization. Future Generation Computer Systems, 67 . pp. 194-205.

Permanent WRAP URL:
http://wrap.warwick.ac.uk/84234

Performance Analysis and Optimization for Workflow Authorization

Cheng Chang[a], Ligang He[b, a], Nadeem Chaudhary[b], Songling Fu[c], Hao Chen[a], Jianhua Sun[a], Kenli Li[a], Zhangjie Fu[d], Ming-Liang Xu[e]
[a]. School of Information Science and Engineering, Hunan University, Changsha, 410082, China
[b]. Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom
[c]. College of Polytechnic, Hunan Normal University, Changsha, China
[d]. School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
[e]. Center for Interdisciplinary Information Science Research, Zhengzhou University, Zhengzhou, China
Email: liganghe@dcs.warwick.ac.uk

***Abstract*—Many workflow management systems have been developed to enhance the performance of workflow executions. The authorization policies deployed in the system may restrict the task executions. The common authorization constraints include role constraints, Separation of Duty (SoD), Binding of Duty (BoD) and temporal constraints. This paper presents the methods to check the feasibility of these constraints, and also determines the time durations when the temporal constraints will not impose negative impact on performance. Further, this paper presents an optimal authorization method, which is optimal in the sense that it can minimize a workflow's delay caused by the temporal constraints. The authorization analysis methods are also extended to analyze the stochastic workflows, in which the tasks' execution times are not known exactly, but follow certain probability distributions. Simulation experiments have been conducted to verify the effectiveness of the proposed authorization methods. The experimental results show that comparing with the intuitive authorization method, the optimal authorization method can reduce the delay caused by the authorization constraints and consequently reduce the workflows' response time.**

I. Introduction

Business processes or workflows are often used to model enterprise applications [1][2][3][4]. A workflow consists of multiple activities or tasks with precedence constraints. When we design workflow management/scheduling strategies, or evaluate the performance of workflow execution on target resources, it is often assumed that when a task is allocated to a resource, the resource will accept the task and start the execution once the processor becomes available. In reality, however, authorization policies may be deployed in the organisations and used to specify who is allowed to perform which tasks at what time. When these authorization schemes are taken into account, the situation can become complex.

A number of authorisation schemes have been presented in [5][6][7]. The RBAC (Role Based Access Control) scheme is one of most popular authorisation schemes. Under the RBAC scheme, users are assigned to certain roles while the roles are associated with prescribed permissions. Therefore, the organisations can control the users permissions through these roles. The following example in banking illustrates the effect of the RBAC scheme on the workflow execution [8]. A bank often uses a variety of computing applications to support its business; these applications may be deployed in a central resource pool (e.g., a cluster) of the bank. A workflow may consist of tasks such as credit data checks, automated signature approval, risk analysis and so on. In each task, a particular application has to be launched to perform the corresponding business functions. Under RBAC, an application may only be launched by certain users (i.e., the employees in the bank) assuming certain roles (i.e., job positions, such as branch manager or financial advisor). The following authorisation constraints are often encountered in such scenarios: 1) Role constraints: A task may only be performed by a particular role; 2) Temporal constraints: A role or a user is only activated during certain time intervals (e.g., a staff member only works in certain hours of a day); 3) Separation of Duty constraints: If Task A is run by a role (or a user), then Task B must not be run by the same role (or user); 4) Binding of Duty constraints: If Task A is run by a role (or user), then Task B must be run by the same role (or user). Since a valid and activated role has to be assigned to a task before the task can start execution, these authorisation constraints may delay the start of a task in a workflow, and consequently have negative impact on application performance (e.g. mean response time of workflows). Similar authorization constraints and situation also exist in other application domains such as healthcare systems [9], the manufacturing community [10][11], and other business processes [12][13].

The focus of this paper is to investigate the performance impact of the authorization constraints and the authorization method (i.e., the way of selecting the roles to assign to the tasks). This paper starts with investigating the issue of checking the feasibility of the authorization constraints designed for workload management systems. More specifically, this paper 1) checks whether all tasks in a workflow can be authorized so that the authorization constraints deployed in the system can be satisfied, 2) determines such time durations in which the temporal constraints will not have negative impact on the performance of workflow executions. Then, the methods are developed to quantitatively determine 1) the time duration for the arrivals of the workflows within which the authorization constraints will not have negative impact on the execution performance of the workflows, and 2) the delay caused by the authorization constraints, if a workflow arrives beyond the above duration. Based on the above analyses, this paper further proposes an optimal authorization method under which the delay caused by the authorization constraints can be minimized. The methods of analyzing the authorization behaviour are then extended to handle stochastic workflows, in which the tasks' execution times are not exactly known, but follow certain probability distributions.

Based on the discussions above, it is worth noting the relation between workflow scheduling and authorization method. Workflow scheduling typically refers to deciding the execution order and the resource allocation of workflow tasks, namely, in which order the workflow tasks should be run and which computer node should be allocated to run a particular task. Authorization method refers to deciding which authorization roles should be assumed to run individual workflow tasks. From the processing order, the authorization method is enacted before workflow scheduling. However, if authorization method and workflow scheduling are treated separately, the authorization method may have negative impact on the workflow performance. This is because after the authorization method decides to run a task under a particular role, it is possible that the role is not activated when the task itself is ready to run from the scheduling point of view, namely when the task is at the head of the queue and the allocated computer node becomes available. Consequently, the task has to wait for the assigned role to be activated and its performance is then jeopardized. So a better strategy is that when the authorization method makes the authorization decisions, it takes the scheduling process into account and tries to mitigate the above situation. In order to achieve this, it is necessary to investigate the possible negative impact that the authorization constraints and the authorization method may impose on the workflow execution. This is the motivation and essence of the work presented in this paper.

The rest of this paper is organized as follows. The related work in this topic is presented in II. Section III presents the methods to check the feasibility of role, SoD and BoD constraints deployed in the system. Section IV presents the method to determine the time durations in which the workflow executions will not be delayed by the authorization constraints in the system. Section V presents an optimal authorization method to assign the roles to the tasks in a workflow. Section V also proves the method is optimal in the sense that the method generates the minimal delay caused by the authorization constraints for workflow executions. Section VII concludes the paper.

II. Related work

There is the existing work to check the satisfiability of the authorization constraints in a workflow [14][15][8][16][17]. The work in [15] conducted the theoretical analysis about the satisfiability of the authorization constraints for a workflow. The work conducted theoretical analysis and found out that in order to check whether there is a valid the workflow authorization, it only needs to consider a single linear extension (i.e., a linear ordering) of the tasks in the workflow. There exists a valid workflow authorization if and only if there is also a valid authorization solution for the linear extension. However, the approach proposed in our work is able to obtain all valid authorization solutions. Based on this, our work further develops the authorization methods, aiming to reduce the negative impact imposed by the authorization constraints.

The work in [8] conducts the safety analysis, i.e., analyzes whether a specified authorization state (i.e., the task-role assignments) can be reached under a set of authorization constraints, given an initial authorization state. The work uses the Color Timed Petri Nets (CTPN) to model roles, SoD and temporal constraints, and then converts the constructed CTPN model to an ordinary Petri-Net (PN) model so that the established PN analysis techniques can be applied to generate the results. The work can generate all possible authorization solutions. However, the approach is a bit heavy since it needs to construct the CPTN model, covert the CPTN model to ordinary PN models, and analyze the PN models. In this paper, we model the feasibility checking problem concisely as a Constraint Satisfaction Problem (CSP).

There are also studies to investigate the overhead caused by authorization constraints [18][19]. The work in [18] also applies CTPN to model various authorization constraints, and the interactions between workflow authorization and workflow execution. Then, the work analyzes and obtains the authorization overhead and other associated performance data from the constructed CTPN model. The work makes use of the modelling capability to capture the dynamics in the workflow authorization and execution. The approach is experiment-oriented since the performance data is gathered through running the constructed model in a Petri-Net simulation toolkit, CPN Tools [20]. Also, the CTPN modelling is a heavy approach, and the construction and running of the models could be time consuming. In this paper, we adopt a theoretical approach to analyzing the authorization overhead, and reveals some fundamental properties with regards to authorization overhead (i.e., the delay caused by the authorization constraints). Based on the theoretical analysis of the overhead, this paper further presents an optimal authorization method that is able to minimize the overhead.

III. Checking feasibility of role, SoD and BoD constraints

$\mathbb{S} = \{s_1, \ldots, s_L\}$ denotes the set of services running on the resource pool.

$\mathbb{F} = (T, E)$ denotes a workflow, in which $T = \{t_1, ..., t_N\}$ is a set of tasks in the workflow and $E = \{(t_i, t_j) | t_i, t_j \in T\}$ is a set of directed edges linking task t_i to t_j. A task invokes one of the services in S.

$\mathbb{R} = \{r_1, ..., r_M\}$ denotes the set of roles defined in the authorisation control system. The role constraint specifies the set of roles that are permitted to run a particular service. $\mathcal{C}^r(s_i)$

denotes the role constraint applied to service s_i. $r(s_i)$ denotes the role that is assigned to run s_i. The Separation of Duty (SoD) and the Binding of Duty (BoD) constraint between s_i and s_j are represented as $r(s_i) \neq r(s_j)$ and $r(s_i) = r(s_j)$, respectively.

In this paper, the problem of checking feasibility of role, SoD and BoD constraints is formulated as a Constraint Satisfaction Problem (CSP) [21]. A CSP consists of a triple $< V, D, C >$, where $V = \{v_1, v_2, ..., v_n\}$ is a set of variables, $D = \{D_{v_1}, D_{v_2}, ..., D_{v_n}\}$, where D_{v_i} is the domain of the value of v_i, C is a set of constraints restricting the values that the variables can take. The Feasibility Checking Problem (FCP) in this paper can be modelled as CSP in the following way. The services in FCP are regarded as the variables in CSP. The role constraint of a service is regarded as the domain of the value of the service. The BoD and SoD constraints are regarded as the constraints restricting the values that the services can take.

An example is given below to illustrate the modelling. Assume there are 7 services, $s_1 - s_7$, and 6 roles, $r_1 - r_6$ in the system. The role constraints of service s_i, denoted as $\mathcal{C}^r(s_i)$, are $\mathcal{C}^r(s_1) = \{r_1\}$, $\mathcal{C}^r(s_2) = \{r_2, r_3, r_4\}$, $\mathcal{C}^r(s_3) = \{r_2, r_3, r_5\}$, $\mathcal{C}^r(s_4) = \{r_2, r_3, r_5\}$, $\mathcal{C}^r(s_5) = \{r_2, r_3, r_5\}$, $\mathcal{C}^r(s_6) = \{r_2, r_4\}$, $\mathcal{C}^r(s_7) = \{r_4, r_6\}$. The SoD constraints are $r(t_2) \neq r(t_5)$, $r(t_2) \neq r(t_7)$, $r(t_6) \neq r(t_7)$. The BoD constraints are $r(t_2) = r(t_4)$, $r(t_3) = r(t_5)$. Then the problem of checking feasibility of these authorization constraints can be formulated as CSP as follows.

$CSP =< V, D, C >$,
$V = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$,
$D = \{D_{s_1}, D_{s_2}, ..., D_{s_7}\}$,
$C = \{C_1, C_2, C_3, C_4, C_5\}$,
$D_{s_1} = \{r_1\}$,
$D_{s_2} = \{r_2, r_3, r_4\}$,
$D_{s_3} = \{r_2, r_3, r_5\}$,
$D_{s_4} = \{r_2, r_3, r_5\}$,
$D_{s_5} = \{r_2, r_3, r_5\}$,
$D_{s_6} = \{r_2, r_4\}$,
$D_{s_7} = \{r_4, r_6\}$,
$C_1 : r(t2) = r(t4)$;
$C_2 : r(t2) \neq r(t5)$;
$C_3 : r(t2) \neq r(t7)$;
$C_4 : r(t6) \neq r(t7)$;
$C_5 : r(t3) = r(t5)$;

There are the existing solvers to solve the CSP problem [21]. The solutions are the feasible role assignments to the tasks so that all SoD, BoD and role constraints are satisfied.

IV. Analyzing the coverage of temporal constraints

A. Calculating the coverage of temporal constraints based on exact values of execution times

Roles have temporal constraints. It is useful to check the coverage of roles' temporal availability in a given security setting. We can use the CSP solver to obtain all feasible role assignment solutions for the tasks in a workflow. A denotes the set of all feasible role assignments for the workflow, and $A_k = \{(t_i, r_j)|t_i \in T\}$ denotes the k-th feasible role assignment, in which t_i is a task in the workflow and r_j is the role assigned to t_i. In most cases, a role is activated periodically. For example, the role of bank manager is only activated from 9am to 12pm, and from 2pm to 4pm in a day. Therefore, the temporal constraint of role r_i, denoted as $\mathcal{C}^t(r_i)$ can be expressed as Eq.1, where P_i is the period, $D_i = \{[l_{ij}, u_{ij}]|i \in \mathbb{N}\}$ is the time duration when r_i is activated in the period P_i, and S_i and E_i are the start and end time points when this period pattern begins and ends. E_i can be ∞, meaning the periodic pattern continues indefinitely.

$$\mathcal{C}^t(r_i) = (\mathcal{P}_i, \mathcal{D}_i, \mathcal{S}_i, \mathcal{E}_i) \tag{1}$$

Assume that the exact execution times of the tasks in a DAG (and the scheduling algorithm used to schedule the tasks) are known. Therefore, if we know the arrival time of the entry task in the DAG, we can calculate the start time of every task in the DAG. st_i denotes the start time of task t_i, $r(t_i, A_k)$ denotes the role assigned to task t_i in A_k. Assume $r(t_i, A_k) = r_q$. Assume t_0 is the entry task. Given $A_k = \{(t_i, r_j)|t_i \in T\}$, $\mathcal{C}^t(r(t_0, A_k))$ represents the temporal constraint of the role assigned to t_0. Assume $r(t_0, A_k) = r_p$. $\mathcal{T}(r_q)$ denotes the time durations when $r(t_i)$ has to be activated to run t_i so that t_i can start execution without being delayed by the temporal constraints. Given $\mathcal{C}^t(r_p)$, $\mathcal{T}(r_q)$ can be determined by Eq. 2, where D_j is determined in Eq.3. However, r_q is subject to the temporal constraint, $\mathcal{C}^t(r_q)$. Therefore, The intersection of $\mathcal{T}(r_q)$ and $\mathcal{C}^t(r_q)$, denoted by $I(t_i, A_k) = (P^I_{ki}, D^I_{ki}, S^I_{ki}, E^I_{ki})$, is the time durations when task t_i can start execution immediately without being delayed by the temporal constraints, given a role assignment A_k. P^I_{ki} is $lcm(P_p, P_q)$, i.e., the least common multiple of P_p and P_q. $S^I_{ki} = max(S_p, S_q)$. $E^I_{ki} = min(E_p, E_q)$. Let $D^I_{ki} = \{[l^I_{kij}, u^I_{kij}]|j \in \mathbb{N}\}$.

$$\mathcal{T}(r(t_i, A_k)) = (P_0, D_j, S_0+(st_i-st_0), E_0+(st_i-st_0)) \tag{2}$$

$$D_j = \{[l_{0k} + (st_i - st_0), u_{0k} + (st_i - st_0)]|k \in \mathbb{N}\} \tag{3}$$

As shown above, we calculate $\mathcal{T}(r(t_i, A_k))$ from $\mathcal{C}^t(r(t_0, A_k))$, and then calculate $I(t_i, A_k)$ from $\mathcal{T}(r(t_i, A_k))$. $I(t_i, A_k)$ is a subset of $\mathcal{T}(r(t_i, A_k))$. This means that only when t_0 arrives in a subset of the time durations in $\mathcal{C}^t(r(t_0, A_k))$, t_i's start time falls into $I(t_i, A_k)$. Such a subset of time durations in $\mathcal{C}^t(r(t_0, A_k))$ is called $r(t_0, A_k)$'s effective time durations for t_i in the role assignment A_k, which is denoted by $ET_k(t_0, t_i)$. $ET_k(t_0, t_i)$ can be determined by Eq.4.

$$\begin{aligned} ET_k(t_0, t_i) = \quad & (P_0, \{[l^I_{kij} - (st_i - st_0), \\ & u^I_{kij} - (st_i - st_0)]|j \in \mathbb{N}\}, \\ & S_0, E_0) \end{aligned} \tag{4}$$

We can calculate $ET_k(t_0, t_i)$ for every task t_i in the DAG. $\bigcap_{t_i \in T} ET_k(t_0, t_i)$ is the time durations in $\mathcal{C}^t(r(t_0, A_k))$ that

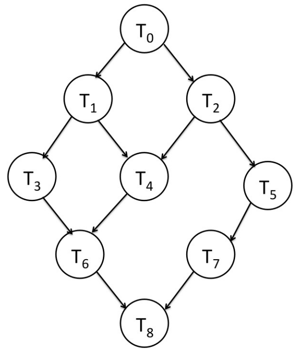

Fig. 1. The workflow in the case study

TABLE I
EXECUTION TIMES OF THE WORKFLOW TASKS IN THE CASE STUDY

Task	Execution time	Task	Execution time
t_0	30	t_1	30
t_2	36	t_3	42
t_4	48	t_5	42
t_6	30	t_7	36
t_8	42		

can ensure the start time of every task $t_i \in T$ $(i \neq 0)$ in the DAG falls into the times durations specified in $\mathcal{C}^t(r(t_i, A_k))$. Only when t_0 arrives in these time durations, can every task in the DAG starts execution without being delayed by the authorization constraints of the role assigned to run the task in A_k. $\bigcap_{t_i \in T} ET_k(t_0, t_i)$ is called t_0's effective arrival time when the role assignment is A_k, denoted by $EA_k(t_0)$. Note that according to the calculation method of $EA_k(t_0)$, $EA_k(t_0)$ is a subset of $\mathcal{C}^t(r(t_0, A_k))$. Therefore, we also call $EA_k(t_0)$ the effective temporal constraint of $r(t_0, A_k)$ for the DAG in the role assignment A_k. Assume $EA_k(t_0) = \{[l_{0j}, u_{0j}] | j \in \mathbb{N}\}$. We can further determine the set of time durations which the start time of t_i falls into, denoted by $EA_k(t_i)$, using Eq.5 . Note that $EA_k(t_i)$ is a subset of $\mathcal{C}^t(r(t_i, A_k))$. Therefore, we call $EA_k(t_i)$ the effective temporal constraint of $r(t_i, A_k)$.

$$EA_k(t_i) = \{[l_{0j} + (st_i - st_0), u_{0j} + (st_i - st_0)] | j \in \mathbb{N}\} \quad (5)$$

We can calculate $EA_k(t_0)$ for every feasible role assignment. Assume $[S, E]$ is the time duration for which we want to check the Coverage of the Temporal Constraints (CTC). If $\bigcup_{A_k \in A} EA_k(t_0)$ cover the entire range of $[S, E]$, then no matter when the workflow instance is initiated, we can always find a role assignment so that all tasks in the workflow can start execution without delay due to the roles' temporal constraints. Otherwise, $[S, E] - \bigcup_{A_k \in A} EA_k(t_0)$ is the time gap during which the execution of at least one task in DAG will be delayed by the current setting of the temporal constraints.

A case study: A case study is given below to illustrate the method of calculating the coverage of the temporal constraints.

Fig. 1 shows the workflow in the case study, in which there are 9 tasks and the execution time of each task is given in Table I.

TABLE II
TEMPORAL CONSTRAINTS OF THE ROLES IN THE CASE STUDY

Role	Temporal	Role	Temporal constraint
r_1	{[09:00, 17:00]}	r_2	{[12:00, 17:00]}
r_3	{[11:00, 17:00]}	r_4	{[09:00, 12:00], [14:00, 17:00]}

TABLE III
ALL FEASIBLE AUTHORIZATION SOLUTIONS IN THE CASE STUDY

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
t_0	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1
t_1	r_3	r_3	r_3	r_3	r_4	r_4	r_4	r_4
t_2	r_1	r_1	r_2	r_2	r_1	r_1	r_2	r_2
t_3	r_1	r_1	r_1	r_1	r_1	r_1	r_1	r_1
t_4	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2
t_5	r_3	r_3	r_3	r_3	r_4	r_4	r_4	r_4
t_6	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2
t_7	r_2	r_3	r_2	r_3	r_2	r_3	r_2	r_3
t_8	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2

There are 4 roles in the system, and the temporal constraint of each role is given in Table II and illustrated in Fig. 2 (for brevity, we assume the temporal constraints of all roles have the same period P of 8 hours, and only show the element D_i in the temporal constraint of a role). Also for simplicity and without compromising the clarity of the illustration, we assume the role constraints are applied to tasks directly in this case study (in the example in Section III, the tasks call one of the services in the system and the role constraints are applied on services).

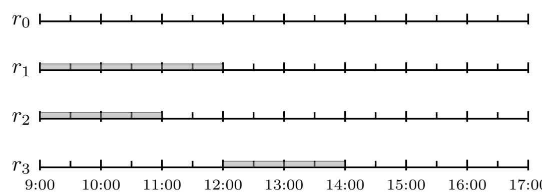

Fig. 2. The temporal constraints of the roles in the case study, the shaded area is the duration when the roles are not activated

Assume that all feasible authorization solutions are as in Table III, after applying the feasibility checking method presented in Section III.

Let us first show how to calculate $EA_1(t_0)$. Since 1) t_0 is authorized to r_1 in A_1, 2) r_1 is activated during $[09:00, 17:00]$, and 3) the execution time of t_0 is 30 minutes (therefore $st_1 - st_0 = 30$ minutes), the possible start time of t_1 can be calculated as below after applying Eq.2, which is also the duration when the role assigned to t_1 in A_1 (i.e., r_3) has to be activated in order for t_1 to start execution without being delayed by the temporal constraints (i.e., $\mathcal{T}(r_3)$).

$$\mathcal{T}(r(t_1, A_1)) = \mathcal{T}(r_3) = \{[09:30, 17:30]\}$$

However, the temporal constraint of r_3 is

$$\mathcal{C}^t(r_3) = [11:00, 17:00].$$

Consequently,

$$I(t_1, A_1) = \mathcal{C}^t(r_3) \bigcap \mathcal{T}(r_3) = \{[11:00, 17:00]\}.$$

TABLE IV
THE VALUES OF $ET_1(t_0, t_i)$ IN THE CASE STUDY

$ET_1(t_0, t_1)$	$\{[10:30, 16:30]\}$
$ET_1(t_0, t_2)$	$\{[09:00, 16:30]\}$
$ET_1(t_0, t_3)$	$\{[09:00, 16:00]\}$
$ET_1(t_0, t_4)$	$\{[10:54, 15:54]\}$
$ET_1(t_0, t_5)$	$\{[09:54, 15:54]\}$
$ET_1(t_0, t_6)$	$\{[10:06, 15:06]\}$
$ET_1(t_0, t_7)$	$\{[10:12, 15:12]\}$
$ET_1(t_0, t_8)$	$\{[09:36, 14:36]\}$

TABLE V
THE VALUES OF $ET_k(t_0)$ IN THE CASE STUDY

$EA_1(t_0)$	$\{[10:54, 14:36]\}$
$EA_2(t_0)$	$\{[10:54, 14:36]\}$
$EA_3(t_0)$	$\{[11:30, 14:36]\}$
$EA_4(t_0)$	$\{[11:30, 14:36]\}$
$EA_5(t_0)$	$\{[14:00, 14:36]\}$
$EA_6(t_0)$	$\{[13:30, 14:36]\}$
$EA_7(t_0)$	$\{[13:30, 14:36]\}$
$EA_8(t_0)$	$\{[13:30, 14:36]\}$

Then,

$$\begin{aligned} ET_1(t_0, t_1) &= \{[11:00 - 30mins, 17:00 - 30mins]\} \\ &= \{[1030, 1630]\} \end{aligned}$$

Similarly, t_2 is authorized to run under r_1 in A_1.

$$\mathcal{T}(r_1) = \{[09:30, 17:30]\},$$

$$I(t_2, A_1) = \mathcal{C}^t(r_1) \bigcap \mathcal{T}(r_1) = \{[09:30, 17:00]\}.$$

Therefore,

$$ET_1(t_0, t_2) = \{[09:00, 16:30]\}.$$

Similarly, $ET_1(t_0, t_i)$ for tasks t_3-t_8 can also be calculated, which are all summarized in Table IV.

Then, the effective arrival time of t_0 (i.e., the arrival time of the workflow), $EA_1(t_0)$, can be calculated as follows.

$$EA_1(t_0) = \bigcap_{t_i \in T} ET_k(t_0, t_i) = \{[10:54, 14:36]\}$$

This means that if the workflow arrives during $[10:54, 14:36]$ and A_1 is used as the authorization solution, all tasks in the workflow can start execution without being delayed by the temporal constraints.

Similarly, we can calculate the value of $EA_k(t_0), (2k8)$ (i.e., other authorization solutions A_2-A_8), which are summarized in Table V.

$$\bigcup_{A_k \in A} EA_k(t_0) = \{[10:54, 14:36]\}$$

This suggests that whenever the workflow arrives in the time duration of [10:54, 14:36], there exists an authorization solution under which all tasks in the workflow can start execution without being delayed by the authorization constraints.

B. Calculating the Probability of Immediate Execution

The derivation in previous section is based on the assumption that the exact execution times of the tasks in a DAG are known (therefore the exact value of st_i, i.e., the start time of task t_i, can be determined). However, in some cases, it is difficult to know the precise execution time of a task in advance. Instead, maybe only the probability distribution of the task's execution time is known. In this subsection, a method is proposed to calculate the probability that all tasks in a DAG can start execution immediately without being delayed by the authorization constraints. We call this probability IEP (Immediate Execution Probability). Essentially, IEP is the probability that the authorization constraints will not pose negative performance impact on the workflow execution.

Assume the execution time et_i of task t_i $(0 \leq i \leq N-1)$ is a random variable. x_i denotes the total execution time of all tasks on the path from t_0 to t_i The completion time ct_i can be expressed as Eq. 6.

$$ct_i = st_i + et_i \tag{6}$$

In a DAG, st_i of task t_i depends on the completion times of all direct predecessors (denoted by $prec(t_i)$). Only after all predecessors of a task are completed, the task becomes ready to execute. Therefore, st_i can be calculated by Eq. 7.

$$st_i = MAX_{t_j \in pred(t_i)} \{ct_j\} \tag{7}$$

From another perspective, st_i can be calculated as the total execution time of all tasks in the longest path from the entry task to t_i in the DAG. Let x_i denote the sum of execution times of all tasks on the longest path from the entry task t_0 to t_i. st_i can be calculated by Eq. 8.

$$st_i = st_0 + x_i \tag{8}$$

1) Maximizing IEP for workflow execution: $r(t_i)$ denotes the role assigned to task t_i in an authorization solution. Assume that the workflow arrives at a time point τ that is within the temporal constraints of the role assigned to task t_0 (i.e., $r(t_0)$). t_0 can start execution immediately without the delay caused by the temporal constraint of $r(t_0)$, namely, $st_0 = l_{0,k}$. Based on Eq. 8, st_i is then $\tau + x_i$ since x_i is the total execution time of the tasks on the longest path from t_0 to t_i. If $\tau + x_i$ is within one of the intervals in the temporal constraint of $r(t_i)$ (i.e., the role assigned to task t_i), t_i can start execution without delay (i.e., the probability that t_i can start execution without delay is 1). Otherwise, the probability is 0.

We now present the method to derive the IEP of the workflow under an authorization solution when a workflow arrives at a certain time point.

Assume a workflow is assigned the roles according to an authorization solution, A_k. When the workflow arrives beyond $\mathcal{C}^t(r(t_0))$, it is impossible that the workflow can start execution immediately. Assume the workflow arrives at a time point τ within $\mathcal{C}^t(r(t_0))$. Then the IEP of task t_i under the authorization A_k can be calculated by Eq. 9, where the value of function $p(\tau + x)$ is 1 when $l_{i,k} \leq \tau + x \leq u_{i,k}$ and is 0 otherwise.

$$IEP_k(t_i, \tau) = \int_0^\infty f(x_i) \times p(\tau + x_i) dx_i \tag{9}$$

We can calculate $IEP_k(t_i, \tau)$ for every task in the workflow. The minimal $IEP_k(t_i, \tau)$ among them all can be regarded as the IEP of the workflow when it arrives at the time point τ and the tasks in the workflow are assigned roles according to the authorization solution A_k, denoted by $IEP_k(\tau)$, which can be expressed as Eq. 10.

$$IEP_k(\tau) = MIN\{IEP_k(t_i, \tau) | 1 \leq i \leq N\} \tag{10}$$

With Eq. 10, we can calculate $IEP_k(\tau)$ for every authorization solution. Namely, we can determine that when a workflow arrives at time τ, the probability that all tasks in the workflow can start execution without delay under any authorization solution. Eq. 11 calculates the maximum value of IEP obtained among all feasible authorization solutions (denoted by $IEP(\tau)$), which can be regarded as the IEP that the workflow can achieve when it arrives at time τ.

$$IEP(\tau) = MAX\{IEP_k(\tau) | A_k \in A\} \tag{11}$$

We can also apply Eq. 10 to calculate such arriving times of the workflow that the IEP of the workflow is no less than a desired value when the workflow is authorized with a particular authorization solution. These arriving times form a time duration in which the workflow can achieve the desired IEP (denoted by IEP_D) under that authorization solution. $I_k(IEP_D)$ denotes such time duration for the authorization solution A_k. Then when the workflow can be assigned to any of the possible authorization solutions, the time durations in which the workflow can achieve IEP_D can be calculated by Eq. 12.

$$I(IEP_D) = \bigcup_{A_k \in A} I_k(IEP_D) \tag{12}$$

If the result of $I(IEP_D)$ in Eq. 12 covers the entire period, then we can conclude that whenever the workflow arrives, there is at least the probability of IEP_D that all tasks in the workflow can start execution without delay caused by the specified authorization constraints.

Eq. 10, 11 and 12 can be utilized to design the authorization method, i.e., to determine the assignment of the authorization solution to an arriving workflow, which is presented in Section V.

Note that we do not specify any particular form of probability distribution for the execution time of workflow tasks. In theory, any probability distribution can be used. However, we can only conduct the mathematical derivation with certain probability distributions to obtain the probability distribution function of $f(x_i)$ in Eq.9. In Subsection IV-B2, we will derive how to derive $f(x_i)$ when the execution times of tasks follow the normal distribution. For the forms of probability distribution that cannot be mathematically derived, we can resort to the numerical methods to calculate the value of Eq. 9.

TABLE VI
THE MEAN AND STANDARD DEVIATION OF THE EXECUTION TIMES OF THE WORKFLOW TASKS IN THE CASE STUDY

Task	μ	σ	Task	μ	σ
t_0	30	3	t_1	30	3
t_2	36	2	t_3	42	3
t_4	48	1	t_5	42	1
t_6	30	2	t_7	36	2
t_8	42	5			

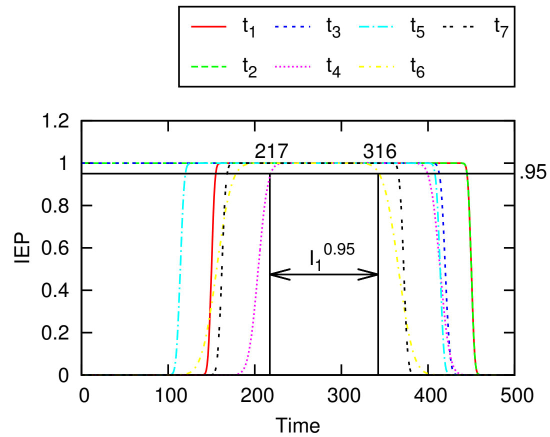

Fig. 3. The IEP result for A_1, the interval [217, 316] is optimal arrival time for workflow

2) A Case Study: To demonstrate the process of calculating the IEPs for the tasks in a workflow, we present a case study assuming that the execution time of a task in a workflow follows a normal distribution and that the expected value and variance of the normal distribution are known. Many real applications and research studies [22], [23], [24] have justified the assumption of normal distribution, which makes the analysis of many random variables tractable analytically. Note that the calculation method does not limit the probability distribution of the tasks' execution time. The execution times can also follow other probability distributions.

The workflow topology in this case study is same as that in Fig. 1. The settings of the temporal constraints in this case study are also same as those in Fig. 2. All feasible authorization solutions are the same as those in Table III.

In Fig. 1, the tasks' execution times have exact values as shown in Table I. In this case study, the execution times of the tasks follow the normal distributions with their means being the same values as those in Table I and but with the deviation being the values in Table VI.

Let $N(\mu, \sigma)$ denote a normal distribution with mean μ and variance σ^2. The execution time of task t_i, et_i, following the normal distribution is expressed by $et_i \sim N(\mu, \sigma^2)$. Before calculating the IEP, we explain two properties of normally distributed random variables.

First, if X_i is normally distributed with expected value μ_i and variance σ_i^2 $(i = 1, 2, 3..., n)$, then $X = \sum_n^i X_i$ is also normally distributed, with the mean of $\sum_n^i \mu_i$ and the variance of $\sum_n^i \sigma_i^2$. Namely, Eq. 13 holds.

$$X \sim N(\sum_n^i \mu_i, \sum_n^i \sigma_i^2). \tag{13}$$

Second, we need to calculate the maximum of a set of random variables in Eq.7. Unfortunately, the maximum value of a set of normal random variable is no longer normally distributed. However, Clark et al. developed a method [25], [26], [27] to recursively estimate the expected value and variance of the maximum value among a finite set of random variables with normal distribution. Based on the estimated expected value and variance, we can obtain the normal distribution which is close to the actual distribution of the random variable defined by the MAX operator. In other words, x_i in Eq. 8 can be approximated as a normal random variable and $f(x_i)$ in Eq. 9 is a normal probability density function (PDF) function.

Fig. 3 depicts the value of IEP of each task (calculated by Eq. 9) as the workflow arrives at different times, being authorized with A_1 in Table III. When the desired value of IEP is set to be 95% (i.e., $IEP_D = 95\%$), we can obtain the interval of the workflow's arrival time for each task in which the value of IEP is no less than 95%. Consequently, we can obtain the interval of the arrival times in which all tasks in the workflow can achieve the IEP of no less than 95% under the authorization A_1 (i.e., $I_1(IEP_D)$), which is $[217, 316]$ as shown in Fig. 3. This result indicates that when the workflow arrives between 217 and 316 seconds, there is at least 95% of probability that when the workflow is authorized with A_1, all tasks in the workflow can start execution without delay caused by the authorization constraints. Similarly, we can obtain $I_k(IEP_D)$ for each A_k. Fig. 4 shows the IEP curves for all possible A_k. $\bigcup_{A_k \in A} I_k(IEP_D)$ (i.e., Eq. 12) is then the interval of the workflow's arrival times in which there is at least 95% of probability that all tasks in the workflow can start execution without delay caused by the authorization constraints.

V. The workflow authorization methods

Section IV-A calculates the time durations when the executions of all tasks in a workflow will not be delayed by the authorization constraints, which is $\bigcup_{A_k \in A} EA_k(t_0)$. The delay caused by the authorization constraints for a task is defined as the time that a ready task (a task in a workflow is ready when all of its predecessors have been completed) has to wait until the role assigned to the task becomes activated. The delay caused by the authorization constraints for a workflow (denoted by td) is defined as the total delay caused by the authorization constraints for the workflow. When a workflow arrives beyond $\bigcup_{A_k \in A} EA_k(t_0)$, it is useful to quantitatively determine td. Further, it is desired to develop an authorization method that can minimize td. This section strives to achieve these.

In this section, we first propose an intuitive policy of authorizing the tasks in a workflow, called the EAF (Earliest Activation First) method. Then we conduct quantitative analysis about the delay caused by temporal constraints. Based on the delay analysis, we further propose a optimized method of authorizing the tasks in a workflow, called the GAA (Global Authorization-Aware) method. The GAA method is optimal in the sense that the method can minimize the delay caused by the temporal constraints. GAA is designed for the scenario where the exact execution times of the workflow tasks are known. In last subsection, we extend the GAA method to work with the stochastic workflows, where we only know the probability distributions of the tasks' execution times.

Algorithm 1. The EAF authorization method

Obtain all feasible authorization solutions using the CSP (Constraint Satisfaction Program) formulation;
for a ready task t_i in the workflow
  Search the set of feasible authorization solutions to obtain a set of roles (denoted by $CA(t_i)$) that can be assigned to t_i;
if all roles in $CA(t_i)$ are not activated,
  Assign to t_i a role with the earliest activation time;
if there are the roles in $CA(t_i)$ that are activated,
  A role is randomly selected and assigned to t_i;

A. The EAF authorization method

The EAF method is intuitive. Its fundamental idea is that when a task in the workflow is ready to run (i.e., all predecessors of the task have completed the executions), but all roles that can be assigned to the task (i.e., satisfy the authorization constraints) are not activated, a role with the earliest activation time will be assigned. The EAF method is outlined in Algorithm 1.

The workflows with different arrival times may have different delay, td, caused by the authorization constraints for a workflow. $td(\tau)$ denotes the delay experienced by the workflow whose arrival time is τ. $td^{EAF}(\tau)$ denotes the delay experienced by all tasks in the workflow whose arrival time is τ when the EAF authorization method is applied.

B. The GAA authorization method

Assume a workflow arrives at time τ. $EA_k(t_0).next(\tau)$ denotes the beginning of the next duration after τ in $EA_k(t_0)$. If the workflow waits for $(EA_k(t_0).next(\tau) - \tau)$, then A_k can be used as the authorization solution of the workflow and the workflow execution can progress without further delay caused by the temporal constraints.

The GAA authorization method is proposed based on the above discussion. The GAA method takes as input the set of all feasible authorization solutions. It finds from the set such a authorization solution, A_k, that A_k generates the minimal value of $(EA_k(t_0).next(\tau) - \tau)$, and authorizes the tasks in the workflow according to A_k. The time complexity of GAA is the number of feasible authorization solutions solved by the CSP (Constraint Satisfaction Problem) formulation, since the authorization methods presented in this paper, no matter

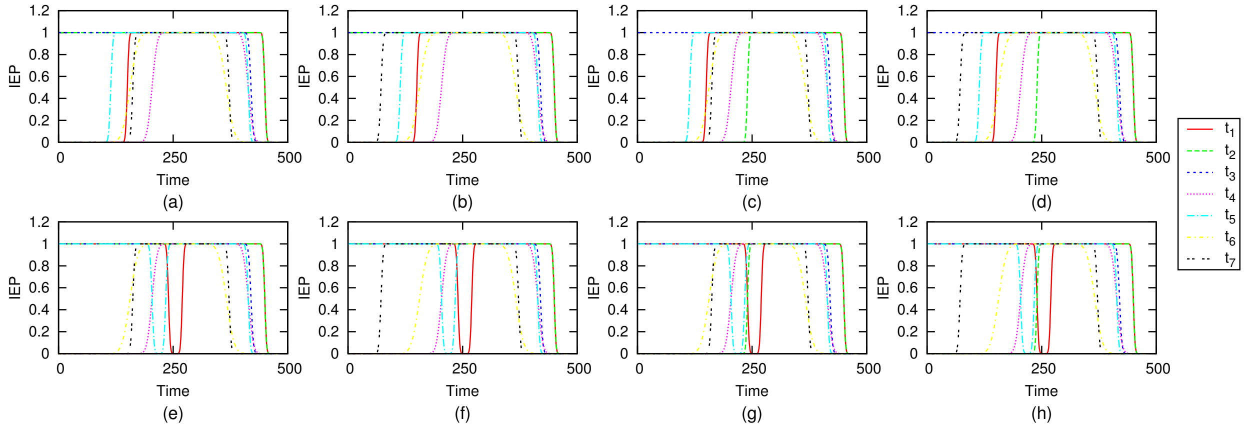

Fig. 4. The IEP curves for all feasible authorizations in the case study.

whether it is EAF or GAA, checks all feasible authorization solutions to find most appropriate one.

Let $td^{GAA}(\tau)$ denote the delay caused by the temporal constraints for a workflow whose arrival time is τ under the GAA method. Then $td^{GAA}(\tau)$ equals to $(EA_k(t_0).next(\tau) - \tau)$. Assume that a workflow arrives at the time point τ, and assume that it turns out that A_k is the authorization solution used for the workflow under the EAF method. We can prove that the delay caused by the temporal constraints for the workflow under the EAF method equals to $(EA_k(t_0).next(\tau) - \tau)$, as shown in Theorem 1.

Theorem 1. *If a workflow arriving at time τ is authorized using the EAF method and the outcome is that the roles are assigned to the tasks in the workflow as in the authorization solution A_k, then Eq.14 holds.*

$$td^{EAF}(\tau) = (EA_k(t_0).next(\tau) - \tau) \tag{14}$$

Proof: If the role assigned to t_0 in A_k (i.e., $r(t_0)$) is only activated at time $EA_k(t_0).next(\tau)$, then t_0 starts execution at $EA_k(t_0).next(\tau)$ under the EAF method. Consequently, the delay caused by the temporal constraints on t_0 is $EA_k(t_0).next(\tau) - \tau$, and according to the definition of $EA_k(t_0).next(\tau)$, all successors of t_0 can start execution without further delay caused by the temporal constraints. Then

$$td^{EAF}(\tau) = (EA_k(t_0).next(\tau) - \tau).$$

Therefore, Eq.14 holds. We call $EA_k(t_0).next(\tau)$ t_0's effective start time (denoted by est_0).

When t_0 starts at $EA_k(t_0).next(\tau)$, we can calculate the start time of t_0's any successor t_i, which is called t_i's effective start time (denoted by est_i) because if t_i starts at time est_i, all successors of t_i can start execution without being delay by the temporal constraints of the roles assigned to the successors in A_k. est_i equals est_0 plus the length of the longest path from t_0 to t_i in the workflow.

If task t_0 starts execution at time τ_0' when the role assigned to t_0 in A_k becomes activated, then the delay caused by the temporal constraints on t_0 is $\tau_0' - \tau$. Assume t_k is t_0's child. If t_0 starts execution at τ_0', then t_k can be ready for execution (t_k's ready time is denoted by τ_k) at time τ_0' plus the length of the longest path from t_0 to t_k (i.e., all its predecessors have been completed), that is, $\tau_0' + (est_k - est_0)$, only subject to the availability of role $r(t_k)$.

If $r(t_k)$ is activated only at est_k, then t_k's delay caused by $r(t_k)$'s temporal constraints is $est_k - (\tau_0' + (est_k - est_0))$=$est_0 - \tau_0'$, and all successors of t_k can start executions without being delayed by temporal constraints. Therefore, $td^{EAF}(\tau)$ can be calculated as

$$\begin{aligned} td^{EAF}(\tau) = &\ (est_0 - \tau_0') + (\tau_0' - \tau) \\ = &\ est_0 - \tau \\ = &\ EA_k(t_0).next(\tau) - \tau \end{aligned}$$

It shows Eq.14 holds.

If $r(t_k)$ is activated at time τ_k' ($\tau_k' < est_k$), then t_k starts execution at τ_k' in the EAF method. We can repeat the analysis similar as above only replacing t_0 with t_k, τ with τ_k and est_0 with est_k. Similarly, we can recursively conduct the analysis for the rest of all tasks in the workflow. It can be shown that Eq.14 holds. ■

Besides the EAF method, other authorization method can be used to assign the roles to the tasks in a workflow. Based on Theorem 1, however, we can prove that no matter what authorization method is used to authorize the workflow, if it turns out that the workflow is authorized as in the authorization solution A_k, then the delay caused by the authorization constraints under the authorization method will be no less than the delay when using the EAF method to assign the roles to the tasks as in A_k. This relation is stated in Theorem 2. The proof of the theorem takes the similar steps as those in Theorem 1. The difference is that when using the EAF method to authorize the tasks as A_k, a task is authorized as soon as the role assigned to the task in A_k becomes activated, while under other authorization method, a task may be authorized (therefore start execution) later than the role's activation time.

Theorem 2. *No matter what authorization method is used to assign the roles to the tasks in a workflow, if the outcome is that the tasks are authorized as in the authorization solution A_k, then the delay caused by the authorization constraints* *under the authorization method is no less than the delay when using the EAF method to authorize the tasks as in A_k.*

Proof: Assume that a workflow arrives at time τ. Similar to Theorem 1, we can determine est_i for every task in the workflow.

If $r(t_0)$ in A_k is activated at time $EA_k(t_0).next(\tau)$, then the minimal delay caused by the authorization constraints is $EA_k(t_0).next(\tau) - \tau$, which equals to the delay generated when using the EAF method to authorize t_0. Any method that authorizes t_0 later than $EA_k(t_0).next(\tau)$ will generate a delay greater than that generated by the EAF method. The theorem holds.

If $r(t_0)$ becomes activated at time τ_0', but under the authorize method, task t_0 is authorized and starts execution at a later time $\tau_0' + \delta_0$ ($\delta_0 > 0$), then the delay caused by the authorization constraints on t_0 is $\tau_0' + \delta_0 - \tau$.

Assume t_k is t_0's child. If t_0 starts execution at $\tau_0'+\delta_0$, then t_k can be ready for execution at time τ_k=τ_0'+δ_0+(est_k-est_0).

Assume $\tau_0' + \delta_0 + (est_k - est_0) \geq est_k$. Then t_k can be authorized and start execution immediately and further, all successors of t_k can be authorized and start execution immediately when they are ready for execution. Therefore, the minimal delay for the workflow is $\tau_0' + \delta_0 - \tau$. Since $\tau_0' + \delta_0 + (est_k - est_0) \geq est_k$, we can have $\delta_0 > est_0 - \tau_0'$. Then the following inequality holds, which shows that the EAF method generates the minimal delay.

$$\begin{aligned}\tau_0' + \delta_0 - \tau > &\quad est_0 - \tau \\ = &\quad EA_k(t_0).next(\tau) - \tau \\ = &\quad td^{EAF}(\tau)\end{aligned}$$

Assume $\tau_0' + \delta_0 + (est_k - est_0) < est_k$. We can repeat the same analysis on t_k as that on t_0, only replacing t_0 with t_k, τ with τ_k and est_0 with est_k. Similarly, we can recursively conduct the analysis for the rest of all tasks in the workflow. It can be shown that the theorem holds.

■

Based on Theorem 1 and 2, we can further prove that the GAA method is the optimal authorization method, as shown in Theorem 3.

Theorem 3. *The GAA authorization method is optimal in the sense that under the GAA method, the delay caused by the authorization constraints for a workflow is not more than that under any other authorization method.*

Proof: Given an authorization method and a workflow arriving at time τ, assume that the method authorizes the tasks as in the authorization solution A_k. From Theorem 2, we know that the delay generated by the authorization method is no less than the delay when using the EAF method to authorize the tasks as in A_k. From Theorem 1, we know that the delay generated by the EAF method can be calculated as $EA_k(t_0).next(\tau) - \tau$. Therefore, the given authorization method generates a delay greater than $EA_k(t_0).next(\tau) - \tau$. According to the algorithm of the GAA method, the GAA method selects the authorization solution A_j that has the least value of $(EA_j(t_0).next(\tau)-\tau)$ from all possible authorization solutions. Therefore, the theorem holds. ■

C. Extending the GAA method to stochastic workflows

The previous subsections present the methods for authorizing the workflows in which the tasks' execution times are exactly known. In this subsection, we extend the GAA method to authorize the worklfow whose constituent tasks have statistically distributed execution times.

When a workflow arrives at time point τ, we apply Eq. 10 and 11 to calculate which authorization solution provides the highest value of $IEP(\tau)$. The calculated authorization solution is then used to authorize the workflow. We call this method the MinIEP method.

In some cases, we want to maximize the opportunity that the arriving workflows can achieve the desired IEP. In order to achieve this, we propose the SGAA (Statistic Global Authorization-Aware) method. In SGAA, we first apply Eq. 10 and 12 to calculate the intervals in which the workflow can acquire the desired IEP_D and also record the corresponding authorization solution that can realize the IEP_D.

Assume a workflow arrives at a time point τ. If τ is within one of the calculated intervals, the workflow is immediately authorized with the corresponding authorization solution. Otherwise, the workflow waits until the start of next nearest interval in $I(IEP_D)$ before it is authorized using the corresponding authorization solution.

VI. Simulation Experiments

We conducted the simulation experiments to evaluate the performance of the GAA method against that of the EAF method. The performance metrics evaluated in the experiments include the delay caused by the authorization constraints for a workflow (i.e., td defined in the first paragraph of Section V) and the response time of a workflow (denoted as *rt*), which is defined as the duration between the time when a first task of the workflow arrives and the time when the last tasks of the workflow is completed.

We also compared the performance among the SGAA, the GAA and the EAF methods. Since SGAA makes authorization decisions based on probability, we adopted the following strategy to compare these three methods. First, a large number of workflow instances are generated with the tasks' execution times following a particular probability distribution. The generated execution times of each task are recorded. We then employ the GAA method to make the authorization decisions for the generated workflow instances. GAA can access the exact execution times of the workflow tasks. GAA is supposed to select the best authorization solution which produces the minimum delay as discussed in Section V-B. Next, we employ the SGAA method to process the same batch of workflow instances. SGAA makes the authorization decisions only based on the probability distributions of the tasks' execution times (without knowing the exact execution times). When SGAA makes the same authorizatoin decision as GAA for a workflow instance (i.e., SGAA makes an optimal authorization decision),

TABLE VII
EXPERIMENTAL SETTINGS

Parameter	Value	Parameter	Value
$TNUM$	15	MAX_DG	3
EX_H	5	$RNUM$	5
MAX_RCST	3	NUM_SoD	4
NUM_BoD	4	P	480
$TEMP$	20%		

we call it a *hit*. We record the proportion of the workflow instances for which the authorization decisions made by SGAA hit, which we call the *hit ratio*. Finally, we employ EAF to process these workflow instances and record its hit ratio. A better authorization method should have a higher hit ratio.

In the experiments, the workflow is randomly generated. Each workflow containing TNUM tasks and each task in a workflow having the maximum of MAX_DG children. RNUM roles are assume to exist in the system. The tasks' role constraints (i.e., the set of roles that a task can assume) are set in the following fashion. The simulation sets a maximum number of roles that any task can assume in the role constraints, denoted as MAX_RCST, which represents the level of restrictions imposed on the role assignment for tasks. When setting the role constraint for task t_i, the number of roles that can run t_i is randomly selected from [1, MAX_RCST], and then that number of roles are randomly selected from the role set.

NUM_SoD and NUM_BoD denote the number of tasks that are associated with SoD and BoD constraints, respectively. Duty constraints were set as follows. Each time, two tasks are randomly selected from the workflow to establish the BoD constraint between them until NUM_BoD tasks are covered. And then the same procedure is applied to establish the SoD constraints among tasks. In this process, the method presented in Section III is used to make sure that the designated duty constraints on these selected tasks can be satisfied. We assume that the tasks execution times follow an exponential distribution. The average execution time of the tasks is the EX_H time units. In order to examine the delay caused by the authorization constraints, a workflow instance is only issued after the previous instance has been completed in the experiments. Unless otherwise stated, the value of *dt* or *rt* depicted in the figure is the value averaged over all workflow instances issued within the period of the temporal constraints, which are set below.

The temporal constraints on roles are set in the following way. For each role, a time duration is selected from a period of *P* time units. The selected time duration occupies the specified percentage of the *P* time units, which is denoted as TEMP. The starting time of the selected duration is chosen randomly from the range of [0, $P \times (1 - TEMP)$]. For example, if *P*=100 and *TEMP*=10%, the starting point is randomly selected from 0 to $90\% \times 100$.

Unless otherwise stated, the parameters are set to be the values shown in Table VII.

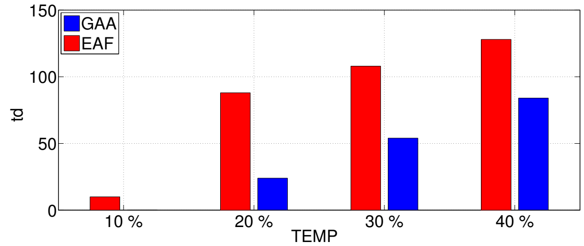

Fig. 5. *td* under different TEMP

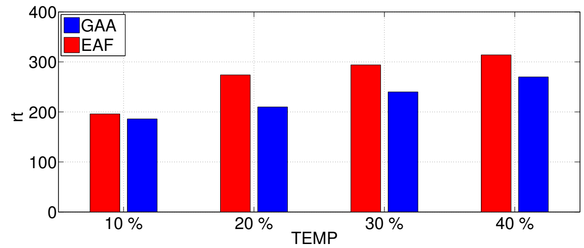

Fig. 6. *rt* under different TEMP

A. Temporal constraints

Fig. 5 shows the change of *td* as the temporal constraints (TEMP) changes. It can be seen from this figure that in all cases the GAA method achieves smaller *td* than EAF. For example, when TEMP is 10%, *td* is 0 under GAA while it is about 10 under EAF. The discrepancy becomes even bigger when TEMP increases. These results verify that the authorization method indeed matters and the GAA method is superior to the intuitive EAF method.

Fig. 6 compares *rt* achieved by GAA and EAF under different TEMP. It can be seen that GAA achieves the shorter *rt* than EAF in all cases. This is because GAA causes less delay and therefore achieves less response time than that under EAF.

B. Arrival times of workflows

The work in this paper presents the method to determine the duration of the time for workflow arrivals within which the authorization constraints will not have negative performance impact. This shows that the arrival time of a workflow has impact on workflow performance. Fig. 7 shows the value of *td* for different workflow arrival times under GAA and EAF. In these experiments, we set the period of all roles (i.e., *P*) as 480 time units, and then issue the workflow instances at the time points from 0 to 300 time units with increment of 60. It can be seen that once again, GAA incurs less *td*than EAF in all cases, except when the arrival time is 300 (whose will be explained later). Further, when the workflows arrive after 120, the GAA method does not cause any delay on workflow executions. These results verify that there indeed exist the durations for the workflow arrivals when the authorization constraints will not delay the workflow executions. The method proposed in this paper is able to theoretically calculate such durations. A point to note is that when the arrival time is 300, no delay is

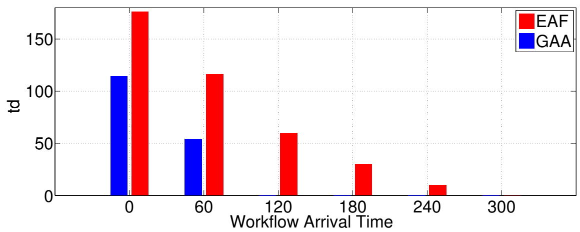

Fig. 7. *td* under different workflow arrival times

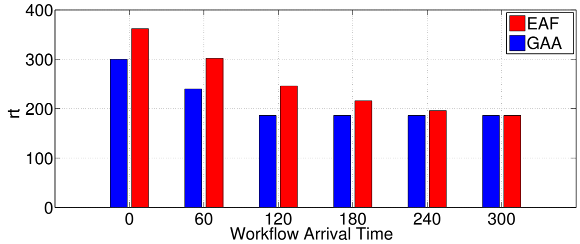

Fig. 8. *rt* under different workflow arrival times

caused under the EAF method either. This is because the time point 300 happens to be within the intersection of $EA_k(t_0)$ of all feasible authorization solutions. Therefore, the system can always find an activated role for any task to enable its execution.

Fig. 8 shows that *rt* of the workflows with different arrival times. Again, GAA outperforms EAF in all cases. The *rt* trend is consistent with the *td* trend shown in Fig. 7.

C. Execution times of the workflow tasks

Obviously, increasing the execution times of the tasks in a workflow will increase the schedule length of the workflow. But do the execution times affect the authorization-related delay? Fig.9 shows the impact of the average execution time of the tasks in a workflow on the coverage of the temporal constraints (CTC), i.e., $\bigcup_{A_k \in A} EA_k(t_0)$. As can be seen from this figure, CTC decreases as the average execution time increases. A reasonable explanation for this is that given a set of temporal constraints, the bigger the execution time of the tasks in a workflow is, the less likely the duration of the workflow execution fits into the temporal constraints. Therefore, CTC may become shorter. This result suggests that given a set of temporal constraints, a workflow with longer tasks may be more likely to be delayed by the temporal constraints that a workflow with shorter tasks, which can be verified by the results presented in Fig. 10.

Fig. 10 demonstrates *td* under different average execution time of workflow tasks. Again, GAA causes less delay than EAF in all cases. It can also be observed from this figure that *td* increases as the average execution time of workflow tasks increases. The results coincides with the results in Fig.9. Indeed, When the execution times increases, CTC decreases. Then more workflow instances issued in the period of the temporal constraints will experience *td*. Consequently, *td*,

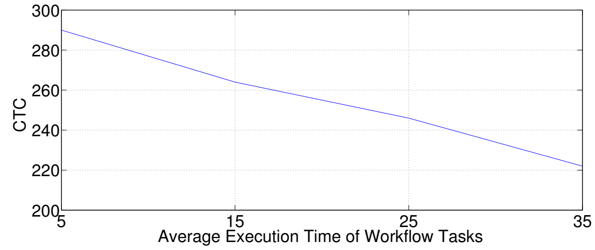

Fig. 9. *rt* under different average execution times of workflow tasks

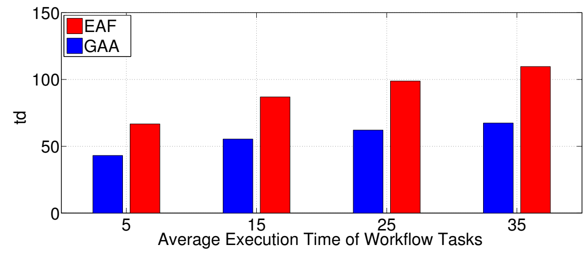

Fig. 10. The coverage of temporal constraints (CTC) under different average execution times of workflow tasks

which is the delay averaged over all workflow instances issued, is bigger.

Fig. 11 shows *rt* generated by the GAA and the EAF method under different average execution time of workflow tasks. As can be observed, the GAA method generates shorter *rt* than EAF in all cases. This again verifies GAA causes less delay than EAF.

D. Hit ratio

In this subsection, we first generate 1000 instances of the workflow in Fig. 1 with the tasks' execution times following the normal distribution. The values of the mean and standard deviation of the distribution for each workflow task are listed in Table VI.

Fig. 12 shows the comparison between SGAA and EAF in terms of the hit ratio. Although The hit ratio curves show the similar trend for the two method, SGAA produces much higher hit count than EAF and in some places (i.e., in the time interval of $[217, 316]$) the hit counts of SGAA is nearly 100%. This result indicates that there are much higher proportion of authorization decisions made by SGAA that are the same as those made by GAA, compared with EAF. As can be seen

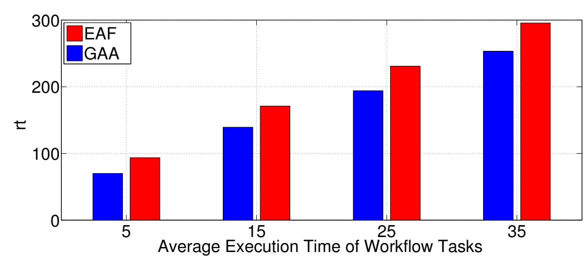

Fig. 11. *rt* under different average execution times of workflow tasks

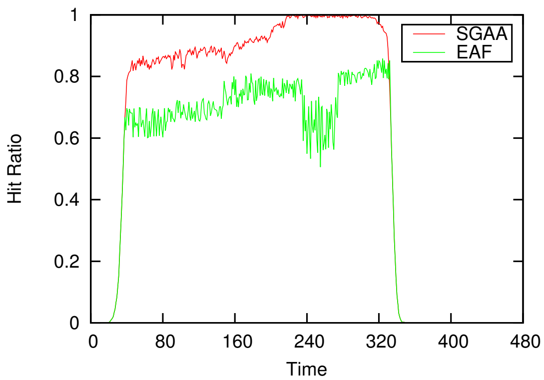

Fig. 12. Comparing the hit ratio between SGAA and EAF

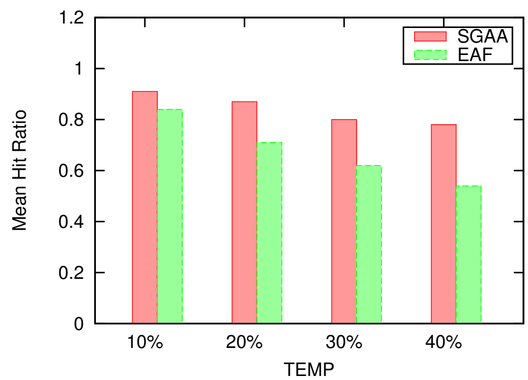

Fig. 13. The hit ratio comparison under different $TEMP$

from the figure, the hit count of EAF becomes unstable in some places (the dip in the EAF curve) and lower than other places. This may be because of the local optimum nature of the EAF method. Namely, some authorization solutions may enter the wrong "branches" of the workflow (e.g., t_1 as shown in our experimental records). In contrast, the performance of SGAA is almost always stable. These experimental results also show that IEP and the optimal interval are effective metrics for measuring the impacts of authorization constraints on workflow executions.

We then change the temporal constraints using the way presented at the beginning of this section. Fig. 13 shows the mean hit ratio achieved by SGAA and EAF under different $TEMP$. It can be seen again that SGAA achieves the higher hit ratio than EAF in all cases. This is because SGAA takes into account the situation of the entire workflow and seek for global optimization and therefore is able to make better decisions than EAF.

VII. Conclusions

This paper investigates the issue of feasibility checking for authorization constraints deployed in workflow management systems. In this paper, the feasibility checking problem is modelled as a constraint satisfaction problem. Further, this paper presents the method to determine the time durations when the deployed temporal constraints do not have negative impact on performance of workflow executions. Moreover, an optimal method is proposed to authorize a workflow, so that the delay caused by the authorization constraints for the workflow executions is minimized. The proposed analysis methods are further extended for the stochastic workflows. The simulation experiments show that the effectiveness of the proposed authorization methods.

VIII. Acknowledgement

The preliminary version of this work has been published in the 20th International Conference on High Performance Computing (HiPC-2013) [28]. This work is partially supported by the Priority Academic Program Development of Jiangsu Higer Education Institutions (PAPD), Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (CICAEET), the Natural Science Foundation of China (NSFC) under Grant Nos. 61472370 and 61672469, and the open project of State Key Laboratory of virtual reality technology and system under Grant No. BUAA-VR-16KF-07.

References

[1] D. Chakraborty, V. Mankar, and A. Nanavati, "Enabling runtime adaptation ofworkflows to external events in enterprise environments," in *Web Services, 2007. ICWS 2007. IEEE International Conference on*, july 2007, pp. 1112 –1119.

[2] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528 – 540, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X08000861

[3] A. Sfrent and F. Pop, "Asymptotic scheduling for many task computing in big data platforms," *Information Sciences*, vol. 319, pp. 71 – 91, 2015, energy Efficient Data, Services and Memory Management in Big Data Information Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025515002182

[4] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Koodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Generation Computer Systems*, vol. 51, pp. 61 – 71, 2015, special Section: A Note on New Trends in Data-Aware Scheduling and Resource Provisioning in Modern {HPC} Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X14002532

[5] G.-J. Ahn and R. Sandhu, "Role-based authorization constraints specification," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 207–226, Nov. 2000. [Online]. Available: http://doi.acm.org/10.1145/382912.382913

[6] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "A generalized temporal role-based access control model," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, no. 1, pp. 4–23, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2005.1

[7] D. Zou, L. He, H. Jin, and X. Chen, "Crbac: Imposing multi-grained constraints on the rbac model in the multi-application environment," *Journal of Network and Computer Applications*, vol. 32, no. 2, pp. 402 – 411, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804508000520

[8] V. Atluri and W. kuang Huang, "A petri net based safety analysis of workflow authorization models," 1999.

[9] M. Stuit, H. Wortmann, N. Szirbik, and J. Roodenburg, "Multi-view interaction modelling of human collaboration processes: A business process study of head and neck cancer care in a dutch academic hospital," *J. of Biomedical Informatics*, vol. 44, no. 6, pp. 1039–1055, Dec. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.jbi.2011.08.007

[10] T. Hara, T. Arai, Y. Shimomura, and T. Sakao, "Service cad system to integrate product and human activity for total value," *CIRP Journal of Manufacturing Science and Technology*, vol. 1, no. 4, pp. 262 – 271, 2009, ¡ce:title¿Life Cycle Engineering¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1755581709000078

[11] J. Y. Choi and S. Reveliotis, "A generalized stochastic petri net model for performance analysis and control of capacitated reentrant lines," *Robotics and Automation, IEEE Transactions on*, vol. 19, no. 3, pp. 474 – 480, june 2003.

[12] D. R. dos Santos, S. E. Ponta, and S. Ranise, "Modular synthesis of enforcement mechanisms for the workflow satisfiability problem: Scalability and reusability," in *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, ser. SACMAT '16. New York, NY, USA: ACM, 2016, pp. 89–99. [Online]. Available: http://doi.acm.org/10.1145/2914642.2914649

[13] C. Bertolissi, D. R. dos Santos, and S. Ranise, "Automated synthesis of run-time monitors to enforce authorization policies in business processes," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. New York, NY, USA: ACM, 2015, pp. 297–308. [Online]. Available: http://doi.acm.org/10.1145/2714576.2714633

[14] J. Crampton, G. Gutin, and D. Karapetyan, "Valued workflow satisfiability problem," in *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '15. New York, NY, USA: ACM, 2015, pp. 3–13. [Online]. Available: http://doi.acm.org/10.1145/2752952.2752961

[15] J. Crampton, "A reference monitor for workflow systems with constrained task execution," in *Proceedings of the tenth ACM symposium on Access control models and technologies*, ser. SACMAT '05. New York, NY, USA: ACM, 2005, pp. 38–47. [Online]. Available: http://doi.acm.org/10.1145/1063979.1063986

[16] Q. Wang and N. Li, "Satisfiability and resiliency in workflow authorization systems," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 4, pp. 40:1–40:35, Dec. 2010. [Online]. Available: http://doi.acm.org/10.1145/1880022.1880034

[17] Y. Lu, L. Zhang, and J. Sun, "Using colored petri nets to model and analyze workflow with separation of duty constraints," *The International Journal of Advanced Manufacturing Technology*, vol. 40, pp. 179–192, 2009, 10.1007/s00170-007-1316-1. [Online]. Available: http://dx.doi.org/10.1007/s00170-007-1316-1

[18] L. He, C. Huang, K. Duan, K. Li, H. Chen, J. Sun, and S. A. Jarvis, "Modeling and analyzing the impact of authorization on workflow executions," *Future Gener. Comput. Syst.*, vol. 28, no. 8, pp. 1177–1193, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2012.03.003

[19] L. He, N. Chaudhary, S. Jarvis, and K. Li, "Allocating resources for workflows running under authorization control," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*, 2012, pp. 58–65.

[20] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured petri nets and cpn tools for modelling and validation of concurrent systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 9, no. 3, pp. 213–254, May 2007. [Online]. Available: http://dx.doi.org/10.1007/s10009-007-0038-x

[21] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," *European Journal of Operational Research*, vol. 119, no. 3, pp. 557 – 581, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221798003646

[22] R. H. Möhring, A. S. Schulz, and M. Uetz, "Approximation in stochastic scheduling: The power of lp-based priority policies," *J. ACM*, vol. 46, no. 6, pp. 924–942, Nov. 1999. [Online]. Available: http://doi.acm.org/10.1145/331524.331530

[23] X. Tang, K. Li, G. Liao, K. Fang, and F. Wu, "A stochastic scheduling algorithm for precedence constrained tasks on grid," *Future Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1083–1091, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.future.2011.04.007

[24] J. Gu, X. Gu, and M. Gu, "A novel parallel quantum genetic algorithm for stochastic job shop scheduling." *Journal of Mathematical Analysis and Applications*, vol. 355, no. 1, pp. 63–81, 2009.

[25] S. C. Sarin, B. Nagarajan, and L. Liao, *Stochastic scheduling. Expectation-variance analysis of a schedule.* Cambridge: Cambridge University Press, 2010.

[26] V. J. Duko Leti, "The distribution of time for clark flow and risk assessment for the activities of pert network structure," *The Yugoslav Journal of Operations Research*, no. 37, pp. 195–207, 2009. [Online]. Available: http://eudml.org/doc/261518

[27] C. E. Clark, "The Greatest of a Finite Set of Random Variables," *Operations Research*, vol. 9, pp. 145–162, 1961.

[28] N. Chaudhary and L. He, "Analyzing the performance impact of authorization constraints and optimizing the authorization methods for workflows," in *Proceedings of the 20th International Conference on High Performance Computing*, ser. HiPC 2013, Bangalore, India, 2013.