

Handling Recoverable Temporal Violations in Scientific Workflow Systems: A Workflow Rescheduling Based Strategy

Xiao Liu¹, Jinjun Chen¹, Zhangjun Wu^{2,1}, Zhiwei Ni², Dong Yuan¹, Yun Yang¹

¹*Faculty of Information and Communication Technologies, Swinburne University of Technology
Hawthorn, Melbourne, Australia 3122*

{xliu, jchen, dyuan, yyang}@swin.edu.au

²*Institute of Intelligent Management, School of Management, Hefei University of Technology
Hefei, Anhui, China 230009*

{wuzhangjun, nzwgd}@hfut.edu.cn

Abstract

Due to the complex nature of scientific workflow systems, the violations of temporal QoS constraints often take place and may severely affect the usefulness of the execution's results. Therefore, to deliver satisfactory QoS, temporal violations need to be recovered effectively. However, such an issue has so far not been well addressed. In this paper, we first propose a probability based temporal consistency model to define the temporal violations which are statistically recoverable by light-weight exception handling strategies. Afterwards, a novel Ant Colony Optimisation based two-stage workflow local rescheduling strategy (ACOWR) is proposed to handle detected recoverable temporal violations in an automatic and cost-effective fashion. The simulation results demonstrate the excellent performance of our handling strategy in reducing both local and global temporal violation rates.

1. Introduction

Scientific workflow systems can be seen as a type of high-level middleware services for high performance computing infrastructures such as peer-to-peer (p2p), grid, or cloud computing [1, 7, 8]. Scientific workflows usually underlie many complex e-science applications such as climate modelling, disaster recovery simulation, astrophysics and high energy physics. In reality, a scientific workflow is normally required to be completed before a specific deadline in order to obtain scientific results on schedule. Therefore, to deliver satisfactory temporal QoS (quality of service), the violations of temporal constraints need to be proactively detected and handled [3]. To date, the work on handling workflow temporal violations is still in its

infancy. Specifically, the two basic requirements are *automation* and *cost-effectiveness*, and thus bring two fundamental problems to be solved.

1) *How to define recoverable temporal violations.* In order to avoid heavy-weight exception handling strategies, temporal violations which are statistically recoverable by light-weight exception handling strategies, are required to be defined upfront. To define recoverable temporal violations, a run-time probability based temporal consistency model is thus required.

2) *How to design automatic and cost-effective exception handling strategies.* As mentioned, heavy-weight handling strategies which involve massive overheads and human interventions are not suitable in practice. To handle recoverable temporal violations, less costly light-weight exception handling strategies need to be designed. However, since the capabilities of light-weight handling strategies are relatively limited compared with their heavy-weight counterparts, it is not realistic to replace heavy-weight handling strategies in all situations. The objective in this paper is to design a light-weight handling strategy which can handle recoverable temporal violations without recruiting additional resources outside the current system.

To address the above problems, this paper proposes a workflow rescheduling based strategy. Firstly, a probability based run-time temporal consistency model is proposed to analyse the probability range where temporal violations can be statistically recovered by light-weight exception handling strategies. Secondly, a novel Ant Colony Optimisation based Two-Stage Workflow Local Rescheduling Strategy (ACOWR.), is proposed to address detected recoverable temporal violations in an automatic and cost-effective fashion. The simulation results demonstrate the excellent

performance of our exception handling strategy in reducing both local and global temporal violation rates.

The remainder of the paper is organised as follows. Section 2 defines the recoverable temporal violations and proposes ACOWR as the exception handling strategy. Section 3 demonstrates the simulation results. Finally, Section 4 addresses the conclusions and points out the future work.

2. Workflow Rescheduling Based Strategy

2.1. Recoverable temporal violations

In this section, we propose a probability based run-time temporal consistency model to define the probability range for recoverable temporal violations. Its build-time counterpart is proposed in [4]. Similarly, we model all activity durations as independent variables that follow the normal distribution model. For statistical analysis, we adopt the “3 σ ” rule which means with a probability of 99.73% that the sample from a normal distribution model is falling into the interval of $(\mu - 3\sigma, \mu + 3\sigma)$ [5]. Therefore, similar to the build-time model, the maximum, mean and minimum durations of activity a_i are defined as $D(a_i) = \mu_i + 3\sigma_i$, $M(a_i) = \mu_i$ and $d(a_i) = \mu_i - 3\sigma_i$ respectively where μ_i is the sample mean and σ_i is the sample standard deviation. Its run time duration is denoted as $R(a_i)$.

Definition: Probability Based Run-Time Temporal Consistency Model. At the run-time execution stage, at activity point a_p where $p \leq l$, the upper bound constraint $U(a_k, a_l)$ with the value of $u(a_k, a_l)$, where $k \leq p$, is said to be of:

1) Absolute Consistency (AC),

$$\text{if } R(a_k, a_p) + \sum_{j=p+1}^l (\mu_j + 3\sigma_j) < u(a_k, a_l),$$

2) Absolute Inconsistency (AI),

$$\text{if } R(a_k, a_p) + \sum_{j=p+1}^l (\mu_j - 3\sigma_j) > u(a_k, a_l),$$

3) $\alpha\%$ Consistency ($\alpha\%$ C),

$$\text{if } R(a_k, a_p) + \sum_{j=p+1}^l (\mu_j + \lambda\sigma_j) = u(a_k, a_l).$$

Here, $U(a_k, a_l)$ denotes an upper bound temporal constraint which covers a_k to a_l . $R(a_k, a_p)$ denotes the sum of runtime durations, λ ($-3 \leq \lambda \leq 3$) is defined as the $\alpha\%$ confidence percentile according to the cumulative normal distribution function of

$$F(\mu_i + \lambda\sigma_i) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\mu_i + \lambda\sigma_i} \frac{e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}}{2\sigma_i^2} \cdot dx = \alpha\%$$

where $0 < \alpha < 100$ [5].

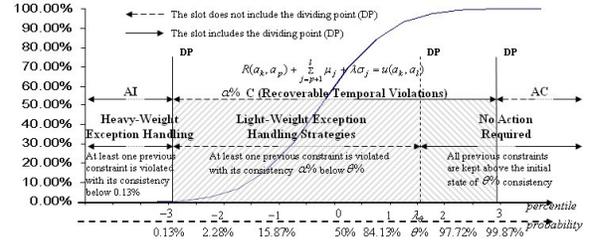


Figure 1. The probability-based run-time temporal consistency model

As depicted in Figure 1, every temporal consistency state is represented with a unique probability value and they together form a continuous Gaussian curve. According to the “3 σ ” rule, the probability range for recoverable temporal violations is defined as (0.13%, 99.87%), represented by the shadowed area. In practice, at scientific workflow run time, exception handling is only triggered in the probability consistency range of (0.13%, $\theta\%$) as shown in the area marked with upwards diagonal lines, while the probability consistency range of ($\theta\%$, 99.87%) marked with downwards diagonal lines requires no action. Here, the threshold of $\theta\%$ denotes the minimum acceptable temporal consistency and it is usually specified through the negotiation between clients and service providers for setting global and local temporal constraints, i.e. temporal QoS contracts. In practice, $\theta\%$ is normally around 90%, i.e. $\mu + 1.28\sigma$, to serve as a reasonable balance point between the client requirement and the system performance. Therefore, apart from AC and AI, all the probability temporal consistency states within the range of (0.13%, $\theta\%$) may produce temporal violations but statistically can be recovered by light-weight exception handling strategies without recruiting additional resources.

2.2. ACOWR

Based on the work in [6] which tackles the workflow scheduling problem with an ant colony optimisation (ACO) algorithm, we propose an novel ant colony optimisation based two-stage workflow local rescheduling strategy (ACOWR). ACOWR is designed to automatically tackle temporal violations through optimising the current plan for task-to-resource assignment. Meanwhile, ACOWR itself only consumes acceptable computation time/cost and does not recruit additional resources outside the current system. Therefore, as will be verified through the simulation experiments demonstrated in Section 3, ACOWR is an effective candidate for handling temporal violations given the requirements of both *automation* and *cost-effectiveness*.

Here, a “two-stage workflow local rescheduling” framework is designed for ACOWR where “two-stage” means a two-stage searching process to strike a balance between the handling of temporal violations and the on-time completion of other workflow instances while “local” means the rescheduling of “local” workflow segments with existing resources. Unlike global rescheduling which modifies the global task-resource list for the entire workflow instance, ACOWR only focuses on the local workflow segment and optimise the integrated task-resource list. Here, the local workflow segment is defined as the set of workflow activities between the next activity of a necessary and sufficient checkpoint (the activity point where a temporal violation occurs) and the end activity of the next local temporal constraint. The integrated task-resource list is an integrated collection of local resources and the integrated DAG task graph which defines the precedence relationships of all the activities in the local workflow segment and their co-allocated activities. Here, co-allocated activities are those which have been allocated to the same resources.

Strategy: ACOWR

Input: Time deficit detected at activity a_p , $TD(a_p)$;
Integrated task-resource list
 $L\{a_i, R_j\} | i = p+1, \dots, p+n, j = 1, 2, \dots, K$;
DAG task graphs $DAG\{G_j | a_j \leq a_n\}$;
Activity duration models $M\{\mu_i, \sigma_i^2\}$;
Resource $R\{R_j, ES(R_j), Cost(R_j) | j = 1, 2, \dots, K\}$.

Output: Rescheduled task-resource list

// Stage 1: Optimising the overall execution time and cost for the integrated task-resource list through ACO algorithm
// Initialisation of pheromone and other parameters for ACO algorithm
1) INITIALISATION(ACO);
2) While (stopping condition is not met)
{
// initialise each ant
for each ant
{
// select heuristics from duration-greedy, cost-greedy and overall-greedy
3) SELECTION(Heuristics);
// build tackling sequence TS based on the input DAG task graphs
4) BUILDING(TS, DAGs);
} }
// construct solutions
5) While (not end of TS)
{
// choose the resource for the next activity based on its earliest start time,
earliest end time, and the bias of B_{ij} (mapping resource R_j to activity a_i)
6) CHOOSE($a_i, a_i, est, a_i, est, R\{R_j\}, B_{ij}$);
// update the est and eet for all the subsequent activities;
7) UPDATE(EST, EET);
// update local pheromone for both duration and cost
8) LOCALUPDATE($d\tau_{ij}, d\tau_{ij}$);
} }
// update the global pheromone based on the makespan and cost of the best-so-far solution
9) GLOBALUPDATE($d\tau_{ij}, d\tau_{ij}, makespan^{best}, cost^{best}$);
// return the best-so-far solution and record into the *SolutionSet*
10) Return(*Solution*, *SolutionSet*)
} }
// Stage 2: Searching for the BestSolution from SolutionSet
11) While (not end of the *SolutionSet*)
{
// compare the compensation time of each *Solution* with the time deficit,
discard the *Solution* if it is smaller than the time deficit
12) COMPARE(*Solution*.et, $TD(a_p)$);
// compare all remained *Solution* and set *BestSolution* as the one with the
minimum cost
13) *BestSolution* = MIN(*Solution*.cost);
} }
14) Return the *BestSolution*;
// return the rescheduled integrated task-resource list and deploy
15) DEPLOY(L)

Figure 2. Algorithm for ACOWR

The pseudo-code for ACOWR is presented in Figure 2. Note that since the algorithm for the first ACO based searching stage is described in [6], here we focus on describing the process without the detailed discussion about ACO and its parameters. The algorithm has five input parameters: the time deficit detected at the checkpoint; the integrated task-resource list; the DAG task graph which define the precedence relationships between tasks; the normal distribution models for activity durations; and resources with their execution speed and the cost per time unit.

As shown in Figure 2, the first searching stage is to optimise the overall execution time and cost for the integrated task-resources list through ACO (Line 1 to Line 10). The ACO algorithm starts from initialisation of pheromone and all parameters (Line 1). In [6], two types of pheromone, i.e. $d\tau_{ij}$ and $c\tau_{ij}$, are defined. Here, $d\tau_{ij}$ denotes the desirability of mapping task a_i to resource R_j from the perspective of execution time while $c\tau_{ij}$ denotes the desirability from the perspective of execution cost. Afterwards, the ACO based searching process iterates until the stopping condition, i.e. the maximum iteration times, is satisfied. In the second searching stage, the *BestSolution* is retrieved from the *SolutionSet* (Line 11 to Line 14). The compensation time of each solution is first compared with the time deficit to filter out unsuccessful solutions (Line 12). Then, the *BestSolution* is defined as the solution with the minimum cost among all the successful solutions (Line 13). Finally, the *BestSolution*, i.e. the solution with the minimum cost and well balanced between the handling of temporal violations and the on-time completion of other workflow instances, is returned as the rescheduled integrated task-resource list (Line 14) and ready for being deployed (Line 15).

3. Evaluation

The simulation experiments are conducted in SwinDeW-G (Swinburne Decentralised Workflow for Grid) which is a peer-to-peer based scientific workflow system running on the SwinGrid (Swinburne service Grid) [7]. All the detailed information such as the program code, experiment settings results can be found online at www.swinflow.org/docs/ACOWR.rar.

In our experiments, the size of scientific workflows ranges from 2,000 to 20,000 based on randomly generated DAG task graphs. Each local workflow segment is composed of 20 to 50 activities and assigned with an upper bound temporal constraint. For the comparison purpose, we record the local and global violation rates under natural situations, i.e. no action

(denoted as NIL), and compared with that of the TDA strategy and our ACOWR. TDA (time deficit allocation) is proposed in [2] which compensates current time deficits by utilising the expected time redundancy.

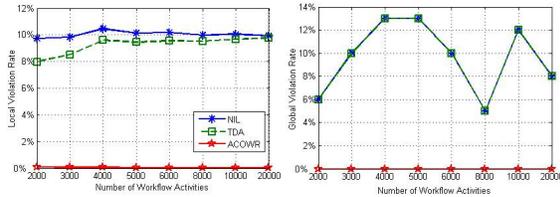


Figure 3. Handling of temporal violations

As can be seen in left subplot of Figure 3, for NIL, the average local violation rate is around 10% since the probability for on-time completion is 90% given the setting of temporal constraints. As for TDA (with an average local violation rate of 9.2%), it can reduce the local violation rates when the size of scientific workflow is small. However, it behaves poorly when the number of activities exceeds 4,000. Since TDA does not compensate time deficits, it cannot delay temporal violations any more when the time deficits accumulates to become large enough. With ACOWR, local violation rate is kept close to 0% (the average local violation rate for ACOWR is 0.04%). The right subplot of Figure 3 shows the results on global violation rates. Since TDA does not compensate time deficits but only delays the local violations, it has no effectiveness on global violations. Therefore, the global violation rates for NIL and TDA are overlapped. The global violation rates for NIL and TDA behave very unstably since NIL and TDA can be easily affected by the dynamic performance of scientific workflow systems. The maximum, minimum and mean global violation rates for NIL and TDA are the same of 13%, 5% and 9.6% respectively. As for ACOWR, the global violation rate is kept very close to 0% since most local temporal violations are handled successfully along workflow execution.

4. Conclusions and Future Work

In this paper, the issue of handling temporal violations in scientific workflow systems has been investigated and addressed by our proposed workflow rescheduling based exception handling strategy. Specifically, a probability based run-time temporal consistency model has been proposed first to identify recoverable temporal violations and then ACOWR, a novel ant colony optimisation based two-stage workflow local rescheduling strategy, has been proposed to automatically tackle those recoverable temporal violations in a cost-effective fashion. The simulation experiments have demonstrated the

excellent performance of our strategy in reducing both local and global temporal violation rates.

In the future, within our “two-stage workflow local rescheduling” framework, some other representative scheduling algorithms such as genetic algorithm and particle swarm optimisation will be investigated and their performance will be compared with ours to determine the best strategy under different scientific workflow system environments.

Acknowledgements

This work is partially supported by Australian Research Council under LP0990393, the National High Technology Research and Development 863 Program of China under Grant No. 2007AA04Z116.

References

- [1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009.
- [2] J. Chen and Y. Yang, "Multiple States based Temporal Consistency for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems," *Concurrency and Computation: Practice and Experience*, Wiley, vol. 19, no. 7, pp. 965-982, 2007.
- [3] J. Chen and Y. Yang, "Temporal Dependency based Checkpoint Selection for Dynamic Verification of Fixed-time Constraints in Grid Workflow Systems", *Proc. In: Proceedings of 30th International Conference on Software Engineering (ICSE2008)*, pp. 141-150, Leipzig, Germany, 2008.
- [4] X. Liu, J. Chen, and Y. Yang, "A Probabilistic Strategy for Setting Temporal Constraints in Scientific Workflows", *Proc. 6th International Conference on Business Process Management (BPM08)*, Lecture Notes in Computer Science, vol. 5240, pp. 180-195, Milan, Italy, Sept. 2008.
- [5] K. A. Stroud, *Engineering Mathematics (Sixth Edition)*. New York: Palgrave Macmillan, 2007.
- [6] Chen W. N. and Zhang J., "An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, no. 1, pp. 29-43, 2009.
- [7] Y. Yang, K. Liu, J. Chen, J. Lignier, and H. Jin, "Peer-to-Peer Based Grid Workflow Runtime Environment of SwinDeW-G", *Proc. 3rd International Conference on e-Science and Grid Computing (e-Science07)*, pp. 51-58, Bangalore, India, Dec. 2007.
- [8] J. Yu, R. Buyya, and K. Ramamohanarao, *Workflow Scheduling Algorithms for Grid Computing, Metaheuristics for Scheduling in Distributed Computing Environments*, F. Xhafa and A. Abraham (eds), ISBN: 978-3-540-69260-7, Springer, Berlin, Germany, 2008.