# QoS Constraint Based Workflow Scheduling for Cloud Computing Services

Guangzhen Lu[1, *], Wen'an Tan[1, 2], Yong Sun[1], Zijian Zhang[1], Anqiong Tang[2]
[1]School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing P.R.China
[2]School of Computer and Information, Shanghai Second Polytechnic University, Shanghai, P.R.China
*Corresponding Author Email: lgzhnuaa@163.com

***Abstract*—QoS constraint based workflow scheduling described by Directed Acyclic Graph (DAG) has been proved to be a NP-hard problem in Cloud Computing Services, especially for cost minimization under deadline constraint. Due to Deadline Bottom Level (DBL) hasn't considered the concurrence during the real executing process that cause much more shatter time to solve such problem, this paper proposes a novel heuristics approach of Concurrent Level based Workflow Scheduling (CLWS). It stratifies all the tasks according to the concurrence among tasks in the actual workflow execution. Then, CLWS distributes the total redundancy time into every level according to their concurrent degree. The simulation experiments show that CLWS makes a better improvement than DBL.**

***Index Terms*—workflow scheduling, cost/time tradeoff, heuristics, concurrent level**

## I. Introduction

With the rapid development of information technology, Cloud Computing is applying to heterogeneous and distributed environment that covers Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [1], which support computers' cooperative work with great efficiency. Workflow, which is regard as an abstract model in computers' cooperative work, plays a more important role in pipeline production of enterprises, office automation, researching and so on. Thus, workflow scheduling is becoming a hot topic. It deals with the allocation of tasks to suitable resources so that the object function can be minimized or maximized while satisfying users' QoS requirements [2]. QoS constraint based workflow scheduling currently includes time minimization under cost constraints and cost minimization under time constraints [3]. This paper focuses on discussion about latter.

To date, many algorithms or strategies, such as Simulated Annealing Algorithm [4], Genetic Algorithm [5] and Hybrid Particle Swarm Algorithm [6], have been proposed to address the problem of QoS constraint workflow scheduling. Although these algorithms can bring an optimized result, time-consuming is still a none-neglected shortage. Thus, many researchers turn to heuristic workflow scheduling [7, 8, 9, 13, 14], and the strategy of workflow task stratifying according to workflow structure and the characteristic of service resources has been got a great attention.

The remainder of this paper is organized as follows. Section II introduces the related work of task stratifying based workflow scheduling. After the problem description in Section III, Section IV explains the proposed workflow scheduling approach. The simulation results are presented in Section V, followed by the conclusion and the future work of this paper.

## II. Related Work

Yu et al. proposed a deadline division strategy named Deadline Min-Cost for scheduling workflow applications with deadline constraints [7]. It divides all the workflow tasks into several levels according to the structure of workflow, and then distributes the deadline into every level. If every level could be completed within the sub-deadline, the entire workflow could also be completed within the overall deadline. Although such approach is simple and can be executed efficiently, some deficiencies still exist: firstly, tasks that can be executed concurrently are not always in the same level; secondly, fixing the time interval of every level will bring much more time pieces. To address above problems of Deadline Min-Cost, Yuan et al. proposed the Deadline Bottom Level denoted as DBL [8]. Diffident with Deadline Min-Cost, DBL stratifies all the tasks by the value of maximal steps to the exit task, and every task's start executing time can be dynamically determined by the actual complete execution time of its parent tasks. From this sense, every task can choose a better service within the extended time interval. Compared with Deadline Min-Cost, although DBL can improve the execution performance of workflow scheduling, some deficiencies are also worthy of paying attention: First, DBL does not stratify tasks under the actual executing situation that more levels are generated, which brings much more time pieces; Second, DBL distributes the overall time float equally to every level, and ignores the differences among levels; Third, when the given deadline is lower than the lower-complete time of workflow for DBL, this algorithm is not suitable.

Inspired by the above two leveling approaches of heuristic workflow scheduling, this paper proposes a novel algorithm called Concurrent Level based


doi:10.4304/jsw.9.4.926-930

Workflow Scheduling (denoted as CLWS). This proposed algorithm stratifies all the tasks according to the concurrence among tasks in the actual executing situation, which can decrease the task levels and increase the utilization of overall time float. Moreover, CLWS distributes the overall time float into every level in the light of differences among levels, which can further increase the utilization of overall time float.

## III. Problem Description

Directed Acyclic Graph (DAG) is typically used to describe the workflow, while the nodes and arcs separately represent the tasks and the relations between tasks of the workflow. For a given workflow, let $G = <V, A>$ denote the DAG, where the symbol $V$ represents the node set, $V = \{1, 2, \ldots, n\}$, and the arc set $A = \{<i, j> | i \rightarrow j$, where $j$ is the direct successor of $i$, and $i, j \in V\}$. There exists time dependence in every arc $<i, j> \in A$, task $j$ can't execute until task $i$ finishes its work. Suppose that every workflow has only one entrance node denoted as $V_{entr}$ and only one exit node denoted as $V_{exit}$, as shown in Fig. 1 [8], task 1 is the $V_{entr}$ and task 16 is the $V_{exit}$.

Each computing service can provide many service levels with differentiated service qualities, i.e., multiple services can provide similar functionality but with different non-functional properties, such as executing time, cost, reliability and so on [2]. Suppose that each task owns a service pool to manage all its services, define $SP(i)$ as the service pool of task $i$, that is $SP(i) = \{ s_i^k = <t_i^k, c_i^k> | 1 \le k \le |SP(i)| \}$, where $|SP(i)|$ represents the service number of service pool $SP(i)$, $t_i^k$ and $c_i^k$ separately represent the executing time and cost of the $k$-th service of task $i$. The service pools of tasks in Fig. 1 are shown in Table 1 [8], to simplify the problem, this paper let $SP(V_{entr}) = SP(V_{exit}) = \{<0, 0>\}$.

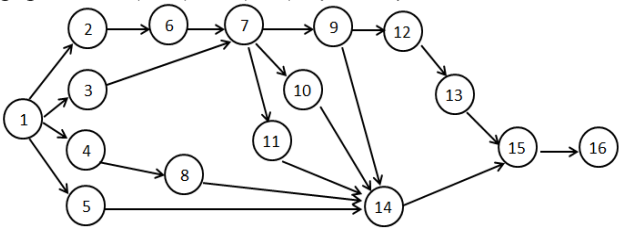


Figure 1. A simple schematic diagram of workflow

TABLE 1.
THE SERVICE POOLS OF TASKS IN FIG. 1

| Task number | Service pool |
|---|---|
| 1 | {<0, 0>} |
| 2 | {<10, 6>, <8, 8>, <6, 11>} |
| 3 | {<5, 10>, <4, 12>} |
| 4 | {<6, 5>} |
| 5 | {<4, 10>, <2, 15>} |
| 6 | {<3, 5>, <2, 10>, <1, 20>} |
| 7 | {<15, 25>, <10, 30>} |
| 8 | {<3, 30>} |
| 9 | {<8, 14>, <5, 18>} |
| 10 | {<30, 100>, <20, 150>, <15, 200>} |
| 11 | {<10, 50>, <6, 80>} |
| 12 | {<9, 18>} |
| 13 | {<25, 40>, <20, 50>} |
| 14 | {<30, 80>, <20, 120>, <15, 150>} |
| 15 | {<13, 50>, <10, 60>} |
| 16 | {<0, 0>} |

The problem of cost minimization under time constraints for workflow scheduling is to let every task select a better service to get the minimal cost according to the given deadline:

$$C_{total} = \min \left\{ \sum_{\substack{i=1 \\ 1 \le k \le |SP_i|}}^{n} c_i^k \right\} \quad (1)$$

To make sure the workflow can execute correctly, several constraint conditions are shown below:

$$\beta_i + t_i^k \le \beta_j \quad (2)$$

$$\delta_n \le t_{deadline} \quad (3)$$

Where $\forall < i, j > \in A$ should satisfy (2), $\beta_i$ represents the start executing time of task $i$, $\delta_n$ represents the sub-deadline of task $n$, and $t_{deadline}$ represents the overall deadline of workflow.

The problem of cost minimization under time constraints for workflow scheduling was proved to be NP-hard [10], and the more time a service is taken, the more cost it should be paid [11]. In order to get the optimized result, CLWS tries to extend every task's time interval $[\beta, \delta]$, and then each task can select a better service with least cost.

## IV. Concurrent Level Based Workflow Scheduling

### A. Relevant Definition

Definition 1. For a given workflow $G = <V, A>$, distributing the minimal time service to each task. By adopting the algorithm of Critical Path to get the critical path, we denote the total executing time of this path as $t_{\min}$, which is the lower-complete time of $G$.

Definition 2. For a given workflow $G = <V, A>$, denote $CP$ as the node set of critical path computed by Definition 1. Assigning the actual start executing time of every task in $CP$ as the earliest start executing time, and allocating tasks, which can be executed concurrently with one critical task (or several critical tasks that have time dependence), into the same level. And make sure that every task's time interval should be within its level's time interval. Detail description is shown as (4), let the concurrent level of $G$ denote as $CL$, and $|CL|$ represents the total number of concurrent levels of the given

workflow.

$$\begin{cases} \beta_{\mathrm{CL}_i} = \min\limits_{\substack{V_k \in CL_i \cap CP \\ 1 \le k \le |CL_i \cap CP|}} \{\beta_{V_k}\} \\ \delta_{CL_i} = \max\limits_{\substack{V_k \in CL_i \cap CP \\ 1 \le k \le |CL_i \cap CP|}} \{\delta_{V_k}\} \\ \beta_{V_i} \ge \beta_{CL_i}, \forall V_i \in CL_i \\ \delta_{V_i} \le \delta_{CL_i}, \forall V_i \in CL_i \end{cases} \tag{4}$$

Definition 3. For a given workflow $G = <V, A>$, in every level, let every single task or tasks that have time dependence defined as *subPath*.

If the last task in one *subPath* hasn't a direct successor in the corresponding level, then define this *subPath*'s sub-deadline as:

$$\delta_{subPath_j} = \delta_{CL_i}, \forall subPath_j \in CL_i \tag{5}$$

If the last task denoted as $V^*$ in one *subPath* exists direct successor, then define this *subPath*'s sub-deadline as:

$$\delta_{subPath_j} = \min_{\substack{V_k \in CL_i \cap succ(V^*) \\ 1 \le k \le |CL_i \cap succ(V^*)|}} \{\beta_{V_k}\}, \forall subPath_j \in CL_i \tag{6}$$

Where $succ(V^*)$ represents the set of direct successors of task $V^*$.

For all the *subPath*s of the given workflow, if $subPath_j$ has unique task $V^*$, then its time interval is set as:

$$\begin{cases} \beta_{V^*} = \max\limits_{\substack{V_k \in CL_i \cap pred(V^*) \\ 1 \le k \le |CL_i \cap pred(V^*)|}} \{\delta_{V_k}\} \\ \delta_{V^*} = \delta_{subPath_j}, subPath_j \in CL_i \end{cases} \tag{7}$$

Where $pred(V^*)$ represents the set of direct predecessors of task $V^*$.

In the same with Deadline Min-Cost [7], if a *subPath* has several tasks, CLWS also uses Markov Decision Process (MDP) [12] to deal with the sequential tasks. It tries to extend every task's time interval, and then each task can select the better service with least cost. The detail description of MDP is shown as below:

Definition 4. For a given workflow $G = <V, A>$, if a *subPath* has several tasks, let *subPath* = {1, 2, …, *r*}, function $f(V_i, S_i^k, V_j)$ represents the process that task *i* selects the *k*-th service and turn to the next task *j* (*j*<*i*). The result value of this function represents the cost of the *k*-th service.

$$f(V_i, S_i^k, V_j) = \begin{cases} c_i^k, & \beta_{V_i} \ge \beta_{subPath} \\ \infty, & others \end{cases} \tag{8}$$

Definition 5. For the *subPath* in the definition 4, MDP uses the equation below to achieve that task $V_i$ selects $S_i^k$, which can make the local cost optimum.

$$F(V_i) = \min_{1 \le k \le |SP_i|} \{ f(V_i, S_i^k, V_j) + F(V_j) \} \tag{9}$$

### *B. The Description of CLWS*

Definition 6. For a given workflow $G = <V, A>$, if the overall deadline is more than the lower-complete time, let their difference as time float denoted as *TF*,

$$TF = t_{deadline} - t_{\min} \tag{10}$$

Distributing the overall time float into every levels under the difference among levels, and every level's sub time float can computed as below,

$$TF_{CL_i} = \frac{TF}{|V| - 2} * |CL_i| \tag{11}$$

From Table 1, it's not useful to distribute time float to both entrance task and exit task. It is obvious that (11) can get a better performance than that of DBL.

After distributing the overall time float, (4) can be modified as below,

$$\begin{cases} \beta_{CL_i} = \min\limits_{\substack{V_k \in CL_i \cap CP \\ 1 \le k \le |CL_i \cap CP|}} \{\beta_{V_k}\} \\ \delta_{CL_i} = \max\limits_{\substack{V_k \in CL_i \cap CP \\ 1 \le k \le |CL_i \cap CP|}} \{\delta_{V_k}\} + TF_{CL_i} \\ \beta_{V_i} \ge \beta_{CL_i}, \forall V_i \in CL_i \\ \delta_{V_i} \le \delta_{CL_i}, \forall V_i \in CL_i \end{cases} \tag{12}$$

Above all, CLWS is explained as below:

Algorithm 1. CLWS

1) Initializing the workflow and allocating services to every task;

2) Computing $t_{\min}$ according to definition 1;

3) Computing the concurrent levels according to definition 2;

4) Computing every level's *subPath*s according to definition 3;

5) If the given overall deadline is more than the upper-complete time, then turn to step 8) and print error message; Otherwise, compute overall time float and distribute them into every level by (12), then compute every level's time interval and its *subPath*s' time interval again;

6) Scanning every *subPath*, if it has unique task, computing its time interval according to (7); Otherwise, adopting MDP to compute this task's time interval;

7) Computing the total cost of this workflow;

8) End.

### *C. An illustrative Experiment*

In order to validate the proposed CLWS, this section adopts data from Fig. 1 and Table 1 to give an illustrative experiment. We suppose that the given overall deadline is 100. After distributing the minimal cost service to every task, the critical path is computed as *CP* = {1, 2, 6, 7, 9, 12, 13, 15, 16} and this workflow's lower-complete time $t_{\min} = 61$.

Combining with the workflow's execution, the Gannt diagram is drawn in Fig. 2, as well as the bold lines represent the executing process of critical task. As shown as this Gannt diagram, all the tasks' actual executing

process, the corresponding concurrent levels and *subPath*s are obviously illustrated.

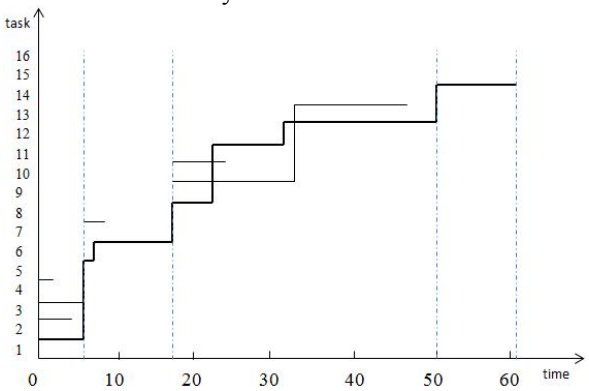


Figure 2. The Gannt diagram of the schematic workflow's executing process

TABLE 2.
THE TASKS' SERVICE SELECTIONS AND ACTUAL TIME INTERVALS OF THE SCHEMATIC WORKFLOW

| Task number | Service selection | Time interval |
|---|---|---|
| 1 | $S_1^1$ | [0, 0] |
| 2 | $S_2^1$ | [0, 10] |
| 3 | $S_3^1$ | [0, 5] |
| 4 | $S_4^1$ | [0, 6] |
| 5 | $S_5^1$ | [0, 4] |
| 6 | $S_6^1$ | [17.14, 20.14] |
| 7 | $S_7^1$ | [20.14, 35.14] |
| 8 | $S_8^1$ | [17.14, 20.14] |
| 9 | $S_9^1$ | [36.5, 44.5] |
| 10 | $S_{10}^1$ | [36.5, 66.5] |
| 11 | $S_{11}^1$ | [36.5, 46.7] |
| 12 | $S_{12}^1$ | [44.5, 53.5] |
| 13 | $S_{13}^1$ | [53.5, 78.5] |
| 14 | $S_{14}^2$ | [66.5, 86.5] |
| 15 | $S_{15}^2$ | [87.20, 97.20] |
| 16 | $S_{16}^1$ | [99.98, 99.98] |

The overall time float *TF* = 100-61 = 39 according to (10). From (11), every level's sub time float is computed, and then corresponding tasks' time interval can be determined. Finally, the schematic workflow's total cost is computed as 493, and all the tasks' service selections and actual executing time intervals are shown in Table 2.

## V. SIMULATION RESULTS

### A. Experiment Environment

All the DAGs are generated by the DAG graph random generator, in which the task numbers $|V| \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$, the task numbers of a subPath, $|subPath| \in \{1, 2, 3, 4\}$, the task's $outDegree \in \{1, 2, 3, 4\}$. The length of service pool can be randomly generated from the interval [5, 10], while the service executing time is randomly generated from the interval [5, 30] and the corresponding cost is inversely proportional to the time. The overall deadline can be computed as $t_{deadline} = t_{\min} + \mu * (t_{\max} - t_{\min})$, where $\mu \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$, $t_{\max}$ is the corresponding upper-complete time with lower-complete time of the workflow and each $\mu$ comprises 10 instances.

### B. Experiment Results and Analysis

This section mainly discusses the executing performances of CLWS, DBL, Deadline Min-Cost and MCP [8]. Suppose that using algorithm *A* and algorithm *B* to execute the same workflow, their total costs are separately denoted as $C_A$ and $C_B$, then the decrease rate from algorithm *A* to algorithm *B* can be denoted as $E_{A|B} = (C_B - C_A)/C_B * 100\%$. Every group adopts the average values.

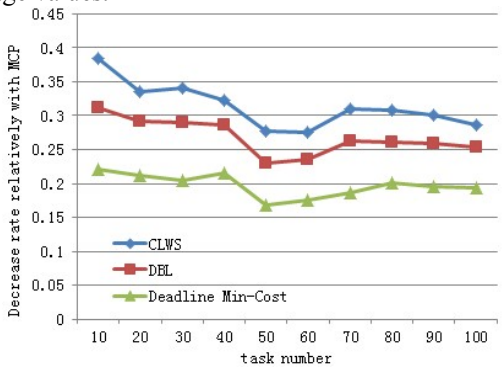


Figure 3. Different algorithms' performance comparison algorithms' total

This part computes the decrement from CLWS, DBL and Deadline Min-Cost to MCP. Fig. 3 displays the compared results. Different with DBL or Deadline Min-Cost, CLWS stratifies all the tasks according to the concurrence among tasks in the actual execute situation, and it can not only decrease the task levels, but also increase the utilization of overall time float. And we can further increase the utilization of overall time float by adopting the approach of distributing the overall time float into every level according to the differences among levels. From Fig. 3 we can also obviously find that the proposed algorithm CLWS can get a better performance than DBL or Deadline Min-Cost.

## VI. CONCLUSION

This paper proposes a novel heuristic workflow scheduling algorithm CLWS, which it distributes task levels by their concurrence, and adopts the efficiency algorithm MDP to optimize the sequential tasks with time dependency. The contributions of CLWS are that it not only decrease the time pieces, but also can optimize the total executing cost. The experiments' results demonstrate that CLWS has better performance than DBL and Deadline Min-Cost. The future work of this paper is to improve CLWS and address the problem of heuristic workflow scheduling more efficiency in the dynamic Cloud computing environment.

## ACKNOWLEDGEMENT

This paper was supported in part by the National Natural Science Foundation of China under Grant No. 61272036, and key disciplines of Shanghai Second Polytechnic University named Software Engineering under Grant No. XXKZD1301.

## REFERENCES

[1] Sushil Bhardwaj, Leena Jain, Sandeep Jain, "Cloud Computing: A Study of Infrastructure as a Service (IAAS)", IJEIT, 2(1), pp.60-63, 2012.

[2] Yingchun Yuan, Xiaoping Li, "Deadline Division-based Heuristic for Cost Optimization in Workflow Scheduling", Information Sciences, 2562-2575, 2009.

[3] Jia Yu, Rajkumar Buyya, Kotagiri Ramanmohanarao. "Workflow Scheduling Algorithms for Grid Computing", Metaheuristics for Scheduling in Distributed Computing Environments, Springer, pp.173-214, 2008.

[4] Hai Jin, Hanhua Chen, Zhipeng Lu, "QoS Optimizing Model and Solving for Composite Service in CGSP Job Manager", Chinese Journal of Computers, 28(4): 578-588, 2005.

[5] Jianning Lin, Huizhong Wu, "Scheduling in Grid Computing Environment Based on Genetic-algorithm", Journal of Computer Research and Development, 41(12): 2190-2194, 2004.

[6] Mingyuan Yu, Yihua Zhu, Ronghua Liang, "A Grid Service-Workflow Scheduling Using Hybrid Particle Swarm", Journal of Huazhong University of Science and Technology: Natural Science, 36(4): 45-47, 2008.

[7] Jia Yu, Rajkumar Buyya, Tham Chen Khong, "Cost-Based Scheduling of Scientific Workflow Applications on Utility Grids", Proceedings of the 1st IEEE International Conference on e-Science and Grid Computing, IEEE Press, 140-147, 2005.

[8] Yingchun Yuan, Xiaoping Li, Qian Wang, "Bottom Level Based Heuristic for Workflow Scheduling in Grids", Chinese Journal of Computers, 31(2): 283-290, 2008.

[9] Ye Gang, Xianjun Li, Dan Yu, Zhongwen Li, Jie Yin, "The Design and Implementation of Workflow Engine for Spacecraft Automatic Testing", Journal of Computers, 6(6): 1145-1151, 2011.

[10] Blythe J, Jain S, Deelman E, "Task Scheduling Strategies for Workflow-based Applications in Grids", Proceedings of the IEEE International Symposium on Cluster Computing and Grid. Cardiff, Vol.2, 759-767, 2005.

[11] Demeulemester E, Herroelen W, Elmaghraby S E, "Optimal Procedures for the Discrete Time/Cost Trade-off Problem in Project Networks", European Journal of Operational Research, 88(1): 50-68, 1996.

[12] RS Sutton, AG Barto, Reinforcement learning: An introduction, MIT Press, USA, 1998.

[13] Guojun Yang, Ying Zheng, Gang Wang, "An Application Research on the Workflow-based Large-scale Hospital Information System Integration", Journal of Computers,6(1): 106-113, 2011.

[14] Chaokun Yan, Huimin Luo, Zhigang Hu, Xi Li, Yanping Zhang, "Deadline Guarantee Enhanced Scheduling of Scientific Workflow Applications in Grid", Journal of Computers, 8(4): 842-850, 2013.