J Supercomput (2015) 71:3811–3831
DOI 10.1007/s11227-015-1469-x

Adaptive dual-criteria task group allocation for clustering-based multi-workflow scheduling on parallel computing platform

Ying-Lin Tsai[1] · **Hsiao-Ching Liu**[1] · **Kuo-Chan Huang**[1]

Published online: 30 June 2015

Abstract Workflow scheduling has long been an important research topic in the field of parallel computing. Clustering-based methods are one of the major types of workflow scheduling approaches which have been shown superior to other kinds of methods in many cases due to their advantage of minimizing inter-task communication costs. Most previous research dealt with single workflow scheduling and focused on how to cluster the tasks within a workflow into a set of task groups. Recent research showed that utilizing idle time gaps between scheduled tasks is a promising direction for efficient multiple workflow scheduling. Since executing multiple workflows simultaneously is an inevitable need in modern shared parallel computing platforms, efficient task group allocation becomes a critical issue. In this paper, we studied such issue and proposed an innovative dual-criteria task group allocation method which considers both task group's finish time and potential resource utilization to effectively improve overall multi-workflow execution performance. In addition, an adaptive task group rearrangement mechanism was adopted to further improve performance. The proposed method has been evaluated with a series of simulation experiments and compared to previous approaches. The experimental results show that our method outperforms previous approaches across different workload conditions and workflow properties in terms of average makespan. The performance improvement ranges from 5 to 29 % for different con-

✉ Kuo-Chan Huang
kchuang@mail.ntcu.edu.tw

Ying-Lin Tsai
jason19890119@gmail.com

Hsiao-Ching Liu
unlimited_19800830@yahoo.com.tw

[1] Department of Computer Science, National Taichung University of Education, Taichung, Taiwan

ditions, achieving the largest performance improvement for workflows of smaller CCR.

Keywords Task allocation · Workflow scheduling · Parallel computing · Clustering-based scheduling

1 Introduction

Parallel task graph scheduling has long been an important research topic in the field of parallel processing and is well known to be a challenging NP-complete problem [1]. With the advancement of technology and emergence of grid and cloud computing, now many large-scale scientific and engineering applications are usually constructed as workflows, whose structure can be represented by traditional parallel task graphs, due to large amounts of interrelated computation and communication [2]. Many open source workflow management systems, such as ASKALON [3], DAGman [4], Gridbus [5], Pegasus [6], have been developed to support workflow applications in parallel and distributed systems.

In general, common workflows usually can be represented by directed acyclic graphs (DAG) [7] for describing the inter-task precedence constraints. Figure 1 is an example of such kind of workflows. Each node represents a task which executes a specific program. The number next to each node means the required execution time of the task. The edges represent the dependence between tasks and the number next to an edge means the inter-task data transmission time. A workflow scheduler has to schedule and allocate each task according to the dependence specified in the workflow definition.

In practice, as discussed [2], DAGs of fork–join control structures are the most common type of underlying structures for many workflow applications. Figure 2 is an example of such kind of workflow structure. There are languages and middleware, such

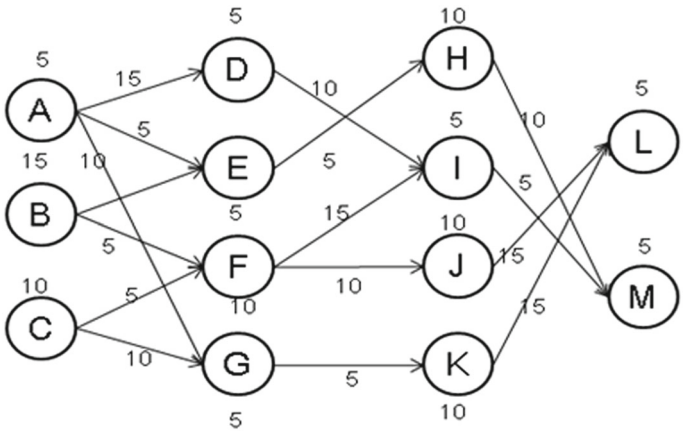


Fig. 1 General DAG-based workflow

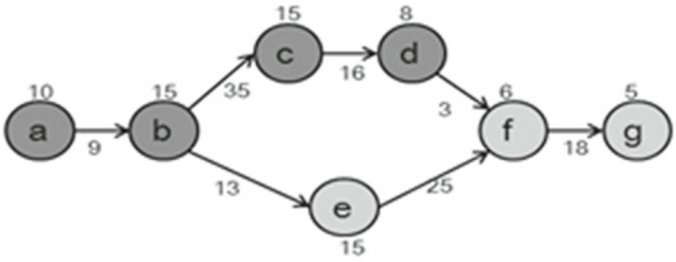


Fig. 2 A workflow example of fork–join structure

as BPEL [8] and Xavantes [9], developed for programming such kinds of workflow applications.

Many approaches have been proposed to deal with the challenging workflow scheduling problem in the literature [10–18]. Clustering-based methods are one of the major types of workflow scheduling approaches and have the advantage of minimizing inter-task communication costs, which makes them superior to other kinds of methods in many cases. Due to the complexity, most previous workflow scheduling research focused on scheduling a single workflow on parallel systems [10–15]. However, as modern high-performance computing platforms, such as grid and cloud, become prevalent, many users would run their workflow applications simultaneously on the same platform. It becomes an inevitable issue to schedule multiple concurrent workflows efficiently. Although most previous researches on clustering-based workflow scheduling focused on the task clustering issue, recent research [19] showed that task allocation utilizing idle time gaps between scheduled tasks is a promising direction for efficient multiple workflow scheduling.

Most previous task allocation approaches adopted simple heuristics which focused on a single principle, e.g., best resource fitness or earliest finish time (EFT). In this paper, we study the issue of task group allocation for clustering-based multi-workflow scheduling and make contributions including proposing an efficient dual-criteria task group allocation method and analyzing the relative advantage of the best-fit and EFT principles across different workload conditions and workflow properties. Our method uses a mechanism which considers both resource fitness and tasks' EFT when allocating task groups and can adjust the weights of different principles for adapting to different situations. In addition, an adaptive task group rearrangement mechanism is adopted in our method. These two mechanisms together enable our method to improve the overall multi-workflow execution performance effectively. The proposed method was evaluated with a series of simulation experiments and compared to previous approaches. The experimental results show that our method outperforms previous approaches across different workload conditions and workflow properties in terms of average makespan.

The remainder of this paper is organized as follows. Section 2 discusses related works on workflow scheduling. Section 3 presents our adaptive dual-criteria task

group allocation method. Section 4 evaluates the proposed method and compares it with previous approaches. Section 5 concludes this paper.

2 Related work

List-based and clustering-based approaches are the two major categories of workflow scheduling algorithms. A list-based heuristic approach maintains a list of all tasks of a workflow application according to their priorities and then schedules the tasks based on the list. There are several list-based heuristics proposed in the literature [10–15]. One of the most famous list-based approach is HEFT (Heterogeneous Earliest Finish Time) developed by Topcuoglu et al. [10]. HEFT first computes the rank value of each task based on its computation and communication costs as well as the dependency with other tasks. After that, the tasks are put into a queue in the descending order of the rank value. Then, the scheduler allocates each task in the queue onto the processor which can lead to the earliest finish time for the task. Many existing workflow management systems, e.g., ASKALON [3] and Pegasus [6], apply a HEFT-like heuristic to schedule workflows. In [11], the authors proposed a new task-ranking mechanism and a new task allocation method for the two major steps in list-based workflow scheduling. The proposed task-ranking mechanism is based on the ideas of remaining workload and hybrid ranking, in contrast to the single path-oriented concept widely used in conventional methods. The work [12] is focused on the problem of scheduling more than one workflow at the same time onto a set of heterogeneous resources. The aim is not only to minimize the overall makespan, but also to achieve fairness. A list-based approach called Online Workflow Management (OWM) was proposed [13] for scheduling multiple online mixed parallel workflows. There are four processes in OWM: Critical Path Workflow Scheduling (CPWS), Task Scheduling, Multi-Processor Task Rearrangement and Adaptive Allocation (AA). CPWS process submits tasks into the waiting queue. Task scheduling and AA processes prioritize the tasks in the queue and assign the task with the highest priority to processors for execution. The multi-processor task rearrangement process deals with scheduling holes to improve utilization. Hirales-Carbajal et al. [14] presented an experimental study of deterministic non-preemptive multiple workflow scheduling strategies in a grid. They analyzed scheduling strategies that consist of two and four stages: labeling, adaptive allocation, prioritization, and parallel machine scheduling. Mu et al. [15] presented new list scheduling heuristics with communication contention for workflow applications modeled as DAG. The target platform is a parallel embedded system composed of multiple processors interconnected by buses and/or switches.

The main idea of clustering-based heuristic methods [20] is to reduce communication delay by grouping the tasks of heavy communication into a cluster first and then allocating the cluster of tasks onto the same processor. The Path Clustering Heuristic (PCH) is a typical example of clustering-based heuristics recently proposed [2]. It first uses the clustering technique to generate groups of tasks based on the inter-task dependency. After that, each group of tasks is allocated onto a resource contiguously to minimize the inter-task communication costs. The key advantage

of PCH [2,21] is the reduced communication costs between tasks. Most proposed clustering-based heuristics [2,20,22] focus on different task clustering approaches and pay little attention to the allocation phase, simply based on the EFT principle. However, task allocation becomes a crucial issue when dealing with multiple workflow scheduling. A distributed task group allocation approach was proposed [23] for clustering-based workflow scheduling, which allows the tasks within a group to be allocated among different resources. The proposed approach was shown to outperform traditional task group allocation methods for clustering-based workflow scheduling. However, like previous methods, the approach considers only tasks' EFT.

During task group allocation, there will be some idle time gaps formed on resources because of the inter-task dependency and the data communication costs between different resources. Recent research [19] showed that these idle gaps can be efficiently utilized to improve the performance of multiple workflow scheduling. A best-fit allocation technique was proposed [24] to deal with the task allocation issue for multiple workflow scheduling based on a list-based approach in distributed real-time systems where each job is associated with a deadline for finishing execution. In the approach, the list scheduling heuristic is applied to determine the task allocation sequence. During task allocation, several bin packing techniques, first fit (FF), best fit (BF), and worst fit (WF), are used to search for appropriate idle time gaps. Experimental results show that the best-fit heuristic achieves the best performance in terms of job guarantee ratio [24]. In this paper, our focus is on minimization of average makespan of all workflows. In [25], a gap evaluation method was proposed for task group allocation in clustering-based multiple workflow scheduling, which tried to integrate the BF principle with the consideration of workflow execution time. The gap evaluation method ranks all available idle time gaps in the increasing order of the sum of each gap's start time and its fitness value, where the fitness value is defined to be the difference between the gap size and the size of the task group to be allocated. The idle time gap with the smallest sum will be chosen for task group allocation. The clustering-based workflow scheduling approach based on the proposed gap evaluation method was shown to outperform the BF heuristic [24] in the experiments [25].

Compared to previous task allocation approaches, our adaptive dual-criteria task group allocation method proposed in this paper adopts two innovative mechanisms for clustering-based multiple workflow scheduling. The first is an adjustable idle time gap selection mechanism and the second is an adaptive task group rearrangement mechanism. Based on these two mechanisms, the proposed task group allocation method is expected to further improve the overall multi-workflow execution performance compared to previous approaches.

3 An adaptive dual-criteria task group allocation approach

This section presents our task group allocation approach for clustering-based multi-workflow scheduling, featuring two mechanisms, adjustable idle time gap selection and adaptive task group rearrangement, for improving overall multi-workflow execution

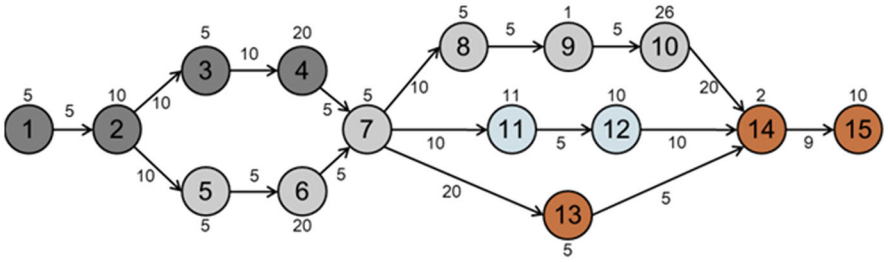


Fig. 3 A task clustering example

performance. These two mechanisms will be described in detail in the following two subsections.

The clustering-based workflow scheduling discussed in this paper, in general, can be divided into three major steps:

1. The first step clusters the tasks in a workflow into several task groups to minimize inter-task communication costs.
2. The second step puts the task groups into the ready queue for allocation according to the priority of each task group. Typical prioritizing mechanisms include the earliest start time (EST) and bottom rank [26] of each task group calculated based on the static information on the task graph.
3. The third step allocates each task group in the ready queue onto an appropriate resource using a specific task group allocation mechanism.

The clustering technique in the first step focuses on reducing the communication costs between tasks. Figure 3 is an example of the typical task clustering technique used in PCH [2,19]. In this example, the clustering process generates four task groups: $\{1, 2, 3, 4\}$, $\{5, 6, 7, 8, 9, 10\}$, $\{11\ 12\}$, and $\{13, 14, 15\}$, as shown with different colors in Fig. 3. In this section, we explore the issues of task group allocation in multiple workflow scheduling based on clustering-based methods and present a new adaptive dual-criteria allocation approach.

3.1 Adjustable idle time gap selection

Most previous clustering-based workflow scheduling approaches focused on how to cluster the tasks in a workflow into different task groups [2]. Although these task groups have to be allocated onto computing resources for execution after the clustering phase, few studies discuss the task group allocation issue. When scheduling workflows onto computing resources, because of inter-task dependency and data communication costs, there are idle time gaps formed between scheduled tasks on each resource. In [24], Stavrinides and Karatza proposed an approach to efficient utilization of the idle time gaps through bin packing techniques. Although, in their approach, the list scheduling heuristic is applied to determine the allocation sequence, the idea can be applied to other kinds of workflow scheduling approaches, too [25]. In the experiments [24], the best-fit (BF) principle was shown to achieve the best overall performance.

Although best-fit allocation has the potential to improve resource utilization, it might delay tasks' start time and in turn degrade the performance of entire workflow because it skips some earlier available time gaps to find the fittest one. Therefore, in our task group allocation method, we adopt an adjustable dual-criteria idle time gap selection mechanism to further improve multiple workflow scheduling performance through making a balance between the task group's finish time and the fitness of an idle time gap. The dual-criteria mechanism defines a score function for evaluating each idle time gap which is large enough to accommodate the task group to be allocated. The score of each time gap is calculated by summing up the EFT of the task group, if allocated on the time gap, and the difference between the lengths of the time gap and the task group. The time gap with the smallest score will be chosen to allocate the task group.

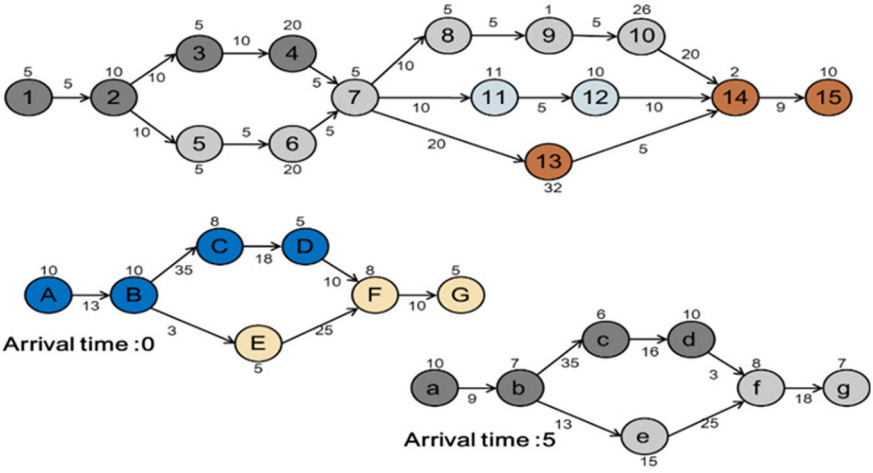
Since the effectiveness of a task group allocation method might be influenced by several different factors, e.g., workflow characteristics and workload conditions, to make the proposed dual-criteria mechanism more flexible for different conditions, we give an adjustable parameter in the score function for adjusting the relative weights of the two different attributes. The score function is defined as follows, where σ is an adjustable parameter ranging between 0 and 1, f is the evaluation of time gap fitness calculated by subtracting the required computation time of the entire task group t_x from the period of the candidate time gap, and the function *EFT( )* calculates the EFT of t_x if allocated on the candidate time gap.

$$\text{Adjustable Allocation}\,(\alpha) = f \times \alpha + (1-\alpha)\,\text{EFT}\,(t_x) \tag{1}$$

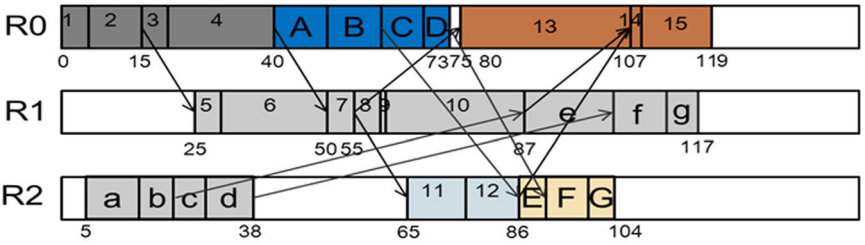
Figure 4 is an example comparing the pure BF principle and our dual-criteria idle time gap selection mechanism. There are three workflows to be scheduled as shown in Fig. 4a. Figure 4b is the schedule produced by the pure BF principle and Fig.4c is the result generated by our dual-criteria idle time gap selection mechanism, where σ is set to 0.5. Figure 4 shows that our dual-criteria mechanism improves the overall workflow execution performance in that the finish times of two workflows get earlier, from 104 to 51 and from 117 to 116, respectively, while the performance of the other one remains the same. Therefore, the average makespan of all the three workflows is reduced from 111.6 to 93.6 as shown in Fig. 4d.

3.2 Adaptive task group rearrangement

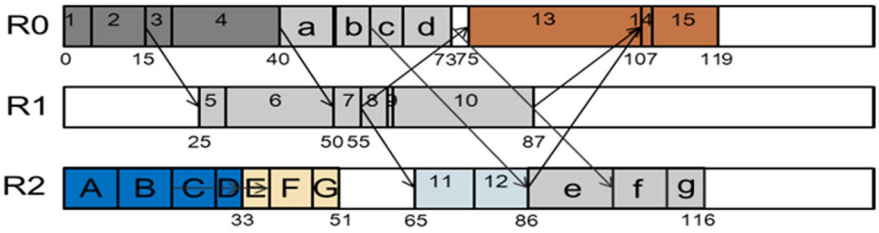
The dual-criteria idle time gap selection mechanism discussed in the previous section tries to allocate an entire task group into a single gap on a specific resource. However, clustering-based workflow scheduling approaches sometimes might lead to task groups too large to fit into any single idle time gap. This, if happening, would result in both degraded resource utilization and delayed task completion time. In the following, we propose an adaptive task group rearrangement mechanism to cooperate with the adjustable dual-criteria idle time gap selection mechanism for further improving the overall multi-workflow execution performance. In traditional clustering-based workflow scheduling approaches, the task groups are formed



(a)



(b)



(c)

	(b)	(c)
average makespan	111.6	93.6

(d)

Fig. 4 Comparison of the pure best-fit heuristic and our dual-criteria mechanism

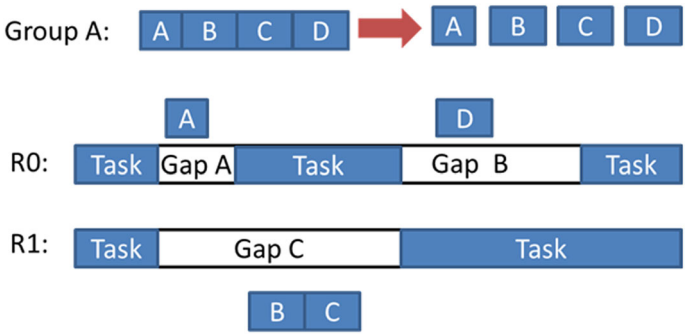


Fig. 5 Individual task allocation in list-based workflow scheduling approaches

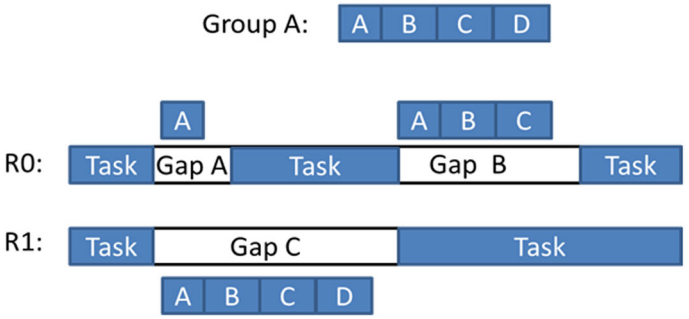


Fig. 6 Advantage of adaptive task group rearrangement

simply based on the workflow properties before the task allocation stage. Our adaptive task group rearrangement mechanism allows a task group to be split into several subgroups for independent allocation at the task allocation stage, to efficiently utilize resources and, in turn, improve the overall workflow execution performance.

Someone might question why not just adopting the list-based workflow scheduling approach instead of allowing a task group in the clustering-based approach to be split into subgroups. Figures 5 and 6 illustrate the potential advantage of our adaptive task group rearrangement mechanism. In Fig. 5, each task is allocated independently as in the list-based workflow scheduling approaches, resulting in some unnecessary inter-task communication overheads. On the other hand, in our adaptive task group rearrangement mechanism, each task group will be cut into subgroups only when necessary at the task allocation stage. At each decomposition activity, an original task group is cut into two new subgroups. The first subgroup contains the largest number of tasks which can be fitted into the gap under consideration, and the other subgroup consists of the remaining tasks. The first subgroup will be allocated first and the second subgroup will be put back to the ready queue, waiting for later allocation decision. Since each subgroup would contain as many tasks as possible, the inter-task communication costs can be minimized. Figure 6

shows the potential advantage of our adaptive task group rearrangement mechanism. The largest subgroup which can be allocated is shown near each idle time gap. In Fig. 6, finally all the four tasks will be allocated in gap C since that leads to the least EFT of the entire task group, resulting in better overall performance compared to Fig. 5. This example shows that our adaptive task group rearrangement mechanism can retain the advantage of clustering-based workflow scheduling to the largest degree while providing additional flexibility for task allocation.

The score function (1) defined in Sect. 3.1 is not appropriate when adopting adaptive task group rearrangement since not every gap can accommodate the entire task group and thus the EFT of the entire task group is not available. To overcome this difficulty, a new score function is defined as follows, where α and β are two adjustable parameters for controlling the relative weights of the three effects and $\alpha + \beta$ ranges between 0 and 1. For idle time gaps which are large enough to accommodate the entire task group, the last term of the score function (2) is zero and the entire score function will become identical to the score function (1) in Sect. 3.1.

$$\begin{aligned}\text{Adaptive Allocation}(\alpha, \beta) = {} & \alpha \cdot f + \beta \cdot EFT \text{ (the first subgroup)} \\ & + (1 - \alpha - \beta) \text{ (communication cost between} \\ & + \text{subgroups computation cost of the second subgroup)}\end{aligned} \tag{2}$$

The following provides an algorithmic description of our adaptive dual-criteria task group allocation approach. The algorithm evaluates each idle time gap in the system in turn according to the above score function in the two nested for loops between lines 1 and 17. Lines 3–5 deal with the case that the gap can accommodate the entire task group. Lines 6–9 handle the case that current gap is not large enough for the entire task group by cutting the task group into two subgroups for allocating the first subgroup first. Lines 11–15 are common to both cases for choosing the most appropriate gap. After the two nested loops, the best gap is found and the task group is split into two subgroups if necessary according to the gap size. The first subgroup is allocated onto the gap and the second subgroup will be put back into the ready queue for later allocation.

Algorithm: Adaptive Dual-Criteria Task Group Allocation

Input:

Tr: total number of resources
n_t: the number of tasks in the task group to be allocated
n_i: total number of gaps on resource i.
α, β: adjustable parameters and $\alpha+\beta$ ranges between 0 and 1
$gap_i(j)$: size of the j_{th} gap on resource i.
$gap_i(j).end$: the end time of the j_{th} gap on resource i
$size_t$: size (total computation cost) of the task group t
$task\_gap_i(j).end$: the expected finish time of the task group if allocated onto the j_{th} gap on resource i without considering the gap size

Variables:

min: the lowest score found so far, initialized as ∞
tempmin: the temporary score of current gap
$final_i.end$: the expected finish time of the task group if allocated onto the last task's finish time on resource i
$final_i$.: the infinite gap starting at the last task's finish time on resource i
i : index of resource.
j : index of gap on resource i.

Output:

found_gap: the index of the gap for allocation
found_res: the index of the resource on which the gap is found
k: index of the decomposition point of the task group to be allocated

```
1.   for i= 1 to Tr do
2.     for j = 1 to ni do
3.       If (gapi(j) ≥ sizet and task_gapi(j).end ≤ gapi(j).end ) then
4.         Tempmin = score according to formula (2) using α and β with the last
                term being zero and the first subgroup equal to the entire
                task group
5.         k = nt
6.       else
7.         According to gapi(j), decompose the task group into two subgroups,
8.         k = the index of the last task in the first subgroup after decomposition
9.         tempmin=score calculated according to formula (2) using α and β
10.      end if
11.      if  (min > tempmin ) then
12.           min = tempmin
13.           found_gap = j
14.           found_res = i
15.      end if
16.    end for
17.  end for
18.  allocate the first subgroup into the jth gap on resource i, and put the
       second subgroup back to the ready queue
```

4 Performance evaluation

In this section, we evaluate the proposed approach through a series of simulation experiments, which compare it with previous methods in the literature in terms of average makespan of all workflows. Here, the makespan of a workflow is defined to

be the time period between its arrival and its finishing execution. We first describe the configurations of the simulation experiments in the following.

4.1 Experimental setting

A DAG generator was implemented to generate workflows of various fork–join structures to be used in the experiments. The simulation experiments were conducted on a PC equipped with an AMD Athlon(tm) 64X Dual Core Processor 5000+ 2.6GHz and 1.87GB RAM. The following settings are used in the simulation experiments.

1. Each workflow contains 20 to 30 nodes randomly.
2. Each workflow has a single entry node and a single exit node.
3. Each workflow contains one to four fork–join structures randomly.
4. Each fork operation produces two to five branches randomly.
5. Each branch contains two to five nodes randomly.
6. Workflows are assumed to be submitted in an online manner with inter-arrival time ranging from 1 to 1,000 s randomly.
7. Each node has the computation cost ranging from 1 to 20 s.
8. Each edge has the communication cost ranging from 1 to 20 s.
9. The DAG generator can generate workflows with different values of communication-to-computation ratio (CCR).
10. We conducted three types of experiments according to different CCR values: 0.1, 1 and 10. Once the CCR value was specified, the computation and communication weights on each node and edge were generated accordingly.
11. The experiments simulated 100 online workflows running on 30 resources.
12. Each experiment was conducted for 30 times and the average value of makespan was calculated.

4.2 Experimental results

This section evaluates the proposed adaptive dual-criteria task group allocation method for clustering-based multiple workflow scheduling. The proposed method is compared with previous approaches, including the best-fit heuristic [24], the EFT heuristic in PCH [2,19], and the EST + fitness approach [25].

Figures 7, 8, and 9 compare the adaptive dual-criteria task group allocation method with previous approaches in terms of average makespan under different CCR values. In the figures, adjustable allocation represents an approach that adopts only the adjustable idle time gap selection mechanism presented in Sect. 3.1, while adaptive dual-criteria is an approach adopting both adjustable idle time gap selection and adaptive task group rearrangement. Next to the name of adjustable allocation is a pair of parentheses indicating the best weight of fitness, i.e., α in score function (1), and the pair of parentheses next to adaptive dual-criteria contains the weights of fitness and EFT, i.e., α and β in score function (2), which lead to the best performance in that case. The inter-arrival time between two consecutive workflows is determined by a random number within a range. In the experiments of Figs. 7, 8, and 9, the range of inter-

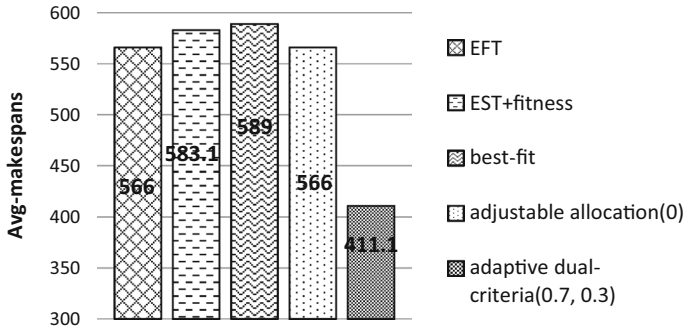


Fig. 7 CCR = 0.1 and inter-arrival time range = 30 s

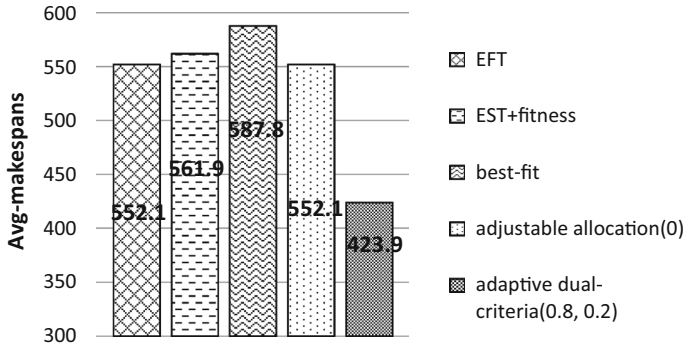


Fig. 8 CCR = 1 and inter-arrival time range = 30 s

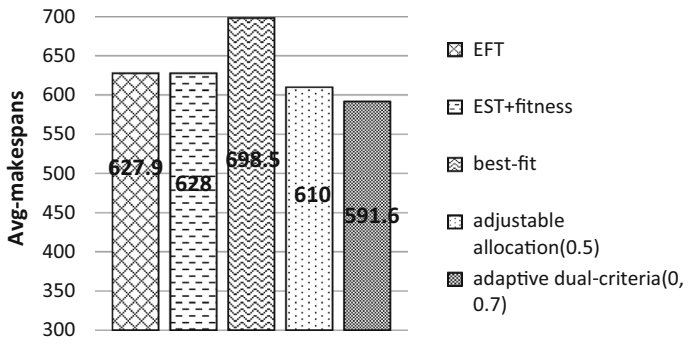


Fig. 9 CCR = 10 and inter-arrival time range = 30 s

arrival time is 30 s. It is obvious that adaptive dual-criteria outperforms the previous approaches and adjustable allocation significantly under all the three different CCR values, while adjustable allocation outperforms EST+fitness and best-fit in all cases but achieves the same performance as EFT when CCR is 0.1 or 1. The experimental results indicate that both adjustable idle time gap selection and adaptive task group rearrangement can improve workflow execution performance and adaptive task group rearrangement is the crucial mechanism for performance improvement.

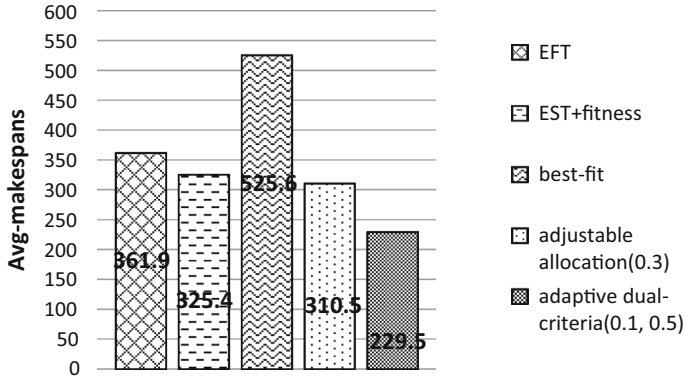


Fig. 10 CCR = 0.1 and inter-arrival time range = 500 s

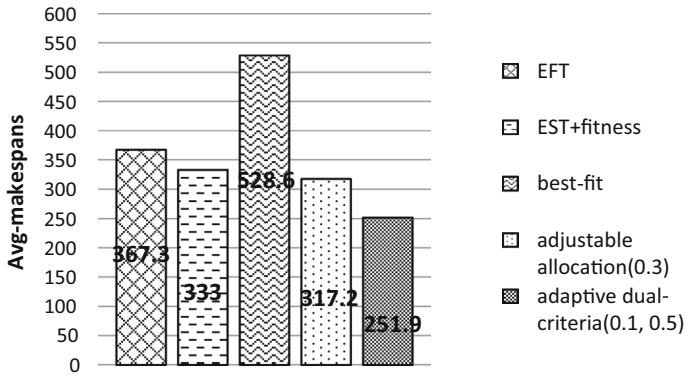


Fig. 11 CCR = 1 and inter-arrival time range = 500 s

For adjustable allocation, the best-fit principle would gain its importance as CCR grows, indicated by the change of α value in the figures, because large CCR implies larger idle time gaps which lead to a higher probability for a left time gap after a task group allocation to be further utilized by later task groups. On the other hand, for adaptive dual-criteria, the best-fit principle is more important than the EFT principle for smaller CCR values, i.e., 0.1 and 1, but becomes less important when the CCR value is high, i.e., 10. This is because in adaptive dual-criteria task groups can be split into smaller subgroups for allocation, raising the utilization of left time gaps after allocation for smaller CCR values. For large CCR value, i.e., 10 in Fig. 9, since there are plenty of large idle time gaps for allocating smaller subgroups, the EFT principle becomes more important than the best-fit principle. In summary, our adaptive dual-criteria approach achieves larger performance improvement compared to previous methods for smaller CCR values because smaller idle time gaps would require special care of task allocation for better resource utilization.

Figures 10, 11, and 12 present the evaluation of the proposed adaptive dual-criteria method when the range of inter-arrival time between workflows is 500 s, much longer

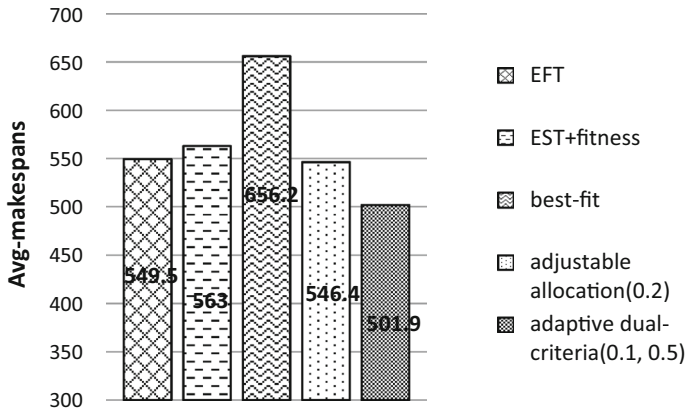


Fig. 12 CCR = 10 and inter-arrival time range = 500 s

Table 1 Count of task group rearrangement

Inter-arrival time range (s)	30			500		
CCR	0.1	1	10	0.1	1	10
Number of task group rearrangement	522	424	188	453	324	137

than in Figs. 7, 8, and 9. Our method still outperforms the previous approaches under all the three CCR values in this set of experiments. Again, the adaptive dual-criteria approach achieves larger performance improvement for smaller CCR values. Since longer inter-arrival time implies fewer task groups compete for the idle time gaps within a fixed time period, the importance of the best-fit principle declines compared to Figs. 7, 8, and 9, as revealed by the α values and the performance difference between EFT and best-fit.

In the above experiments, our method achieves the largest and the least performance improvement when CCR is 0.1 and 10, respectively. Table 1 shows the count of task group rearrangement occurring during the scheduling process for each experiment in the above figures. It reflects that the performance improvement of our method is proportional to the count of task group rearrangement, demonstrating the effectiveness of the adaptive task group rearrangement mechanism. The case that CCR is 0.1 leads to the largest count of task group rearrangement because the idle time gaps are small and most of the task groups have to be cut into subparts for allocation. The count decreases as CCR increases because the idle time gaps are larger for higher CCR values and thus lead to a higher probability of accommodating an entire task group without splitting.

In addition to average makespan, we also evaluated our method with another performance metrics, called win%, which was used for the comparison of different approaches on a per-workflow basis. For each workflow in an experiment, one of the evaluated approaches would lead it to the shortest makespan. The win% value of an approach is defined to be the percentage of the workflows that achieve their shortest

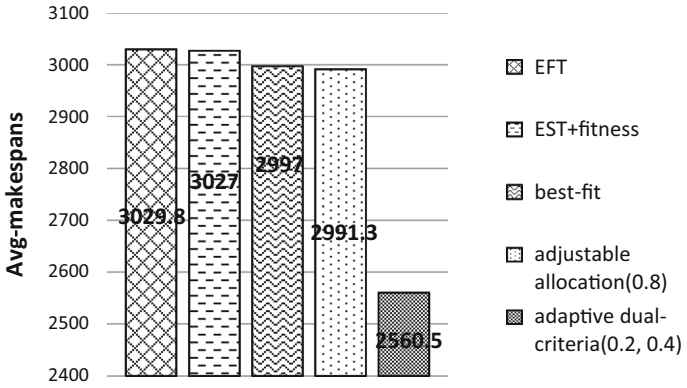


Fig. 13 Inter-arrival time range = 30 s for LIGO

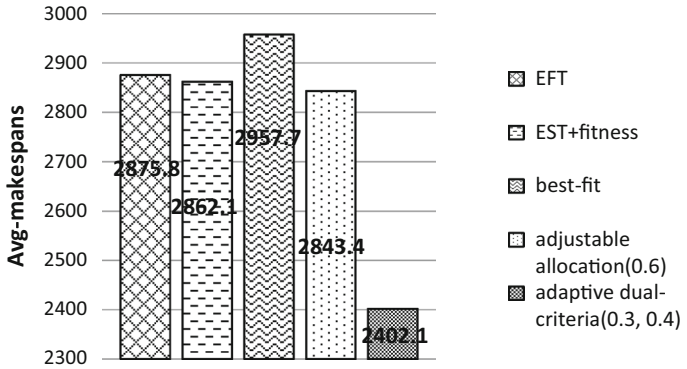


Fig. 14 Inter-arrival time range = 500 s for LIGO

makespan when applying this approach. From users' perspective, a higher win% value represents that an approach has a more stable performance across different workflows and might lead to higher user's satisfaction. For the experiments in Figs. 7, 8, 9, 10, 11 and 12, the win% values of our adaptive dual-criteria task group allocation method are all 100 %.

Figures 13 and 14 show the experimental results based on the structure and properties of a real workflow application, LIGO [27], as shown in Fig. 15. The experimental results show that our adaptive dual-criteria approach can achieve better performance than previous methods [19,21,24] when scheduling workflows of structure and properties like LIGO.

The time complexity of choosing a good gap for task group allocation depends on serval factors, including the number of resources, the number of gaps on each resource in the partial schedule, and the score evaluation function. In our implementation of the methods evaluated in the experiments, each gap would be checked and given a score before determining the best gap for allocation. Therefore, the time complexity of different methods differs mainly in the score function. Among the task group allocation methods evaluated in the experiments, EFT has the lowest time complexity since

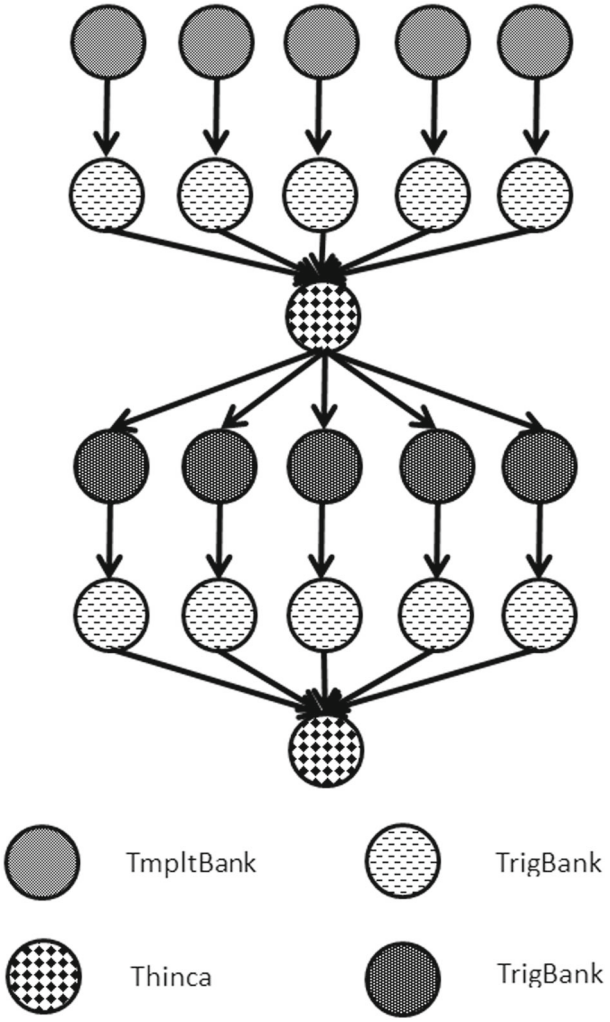


Fig. 15 DAG structure of a real workflow application LIGO

it does not have to calculate the fitness value of each gap. All the other methods, EST+fitness, best-fit, and our adaptive dual-criteria method, have to calculate the fitness value and even perform further computation in the score function, resulting in higher time complexity. Our adaptive dual-criteria method has the most complex score evaluation function, as shown in Eq. (2), and thus has the highest time complexity. Figure 16 compares the scheduling overheads of different methods measured in the experiments. EFT has the smallest overhead. The overheads of EST+fitness, best-fit, and adjustable allocation are very close to each other. Our adaptive dual-criteria task group allocation approach requires the longest computation time. However, compared to the long execution time of scientific workflow applications, usually in hours or even more, the scheduling overhead, in milliseconds, is negligible.

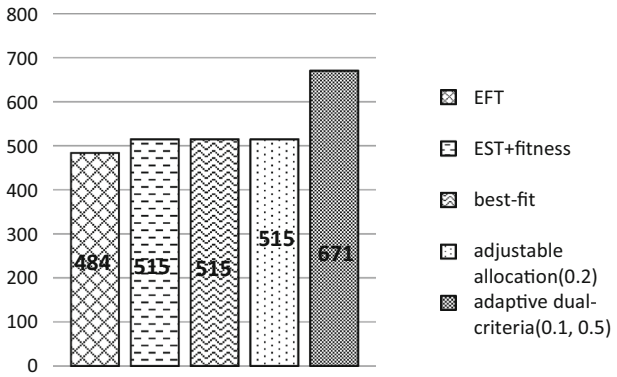


Fig. 16 Scheduling overheads in milliseconds

Table 2 shows resultant average makespan of applying our adaptive dual-criteria task group allocation approach to the set of workflows in Fig. 12 with different α, β values. The results indicate that careful selection of appropriate α, β values is important to achieve good performance. The difference between the best, ($\alpha = 0.1, \beta = 0.5$), and the worst, ($\alpha = 0.9, \beta = 0.1$), performance is enormous. The performance achieved by ($\alpha = 0.9, \beta = 0.1$) is even worse than previous methods according to the data in Fig. 12. Choosing the best α, β values is no easy. Exhaustive experiments, such as Table 2, based on historical workload data can help to find good α, β values for a specific system. The experimental results presented in this section also shed some light on how to choose appropriate α, β values based on information such as CCR values and average inter-arrival time.

5 Conclusion

Workflow scheduling is becoming an important issue as scientific and engineering applications become more complex and computation demanding. Moreover, as large-scale shared high-performance computing platforms, such as grid and cloud, emerge, researchers have to deal with an even more challenging issue of multiple workflow scheduling. Clustering-based methods are one of the major types of workflow scheduling approaches and have been shown superior to other kinds of methods in many cases because of their advantage of reducing inter-task communication costs. Most proposed clustering-based workflow scheduling approaches focused on how to cluster the tasks in a workflow into different task groups, but paid little attention on the issue of task group allocation. In this paper, we propose an adaptive dual-criteria task group allocation method, which takes into account both resource fitness and task groups' EFT when making allocation decisions. In addition to the dual-criteria mechanism, the proposed method also adopts an adaptive task group rearrangement mechanism, which can raise resource utilization effectively, and therefore improve the overall workflow execution performance.

The proposed method was evaluated with a series of simulation experiments and compared to the previous best-fit heuristic in [24], the EFT heuristic [2,19], and the

Table 2 An exhaustive experiment on α, β values

α	β	$1-\alpha-\beta$	Makespans
0.0	0.1	0.9	516.5
0.0	0.2	0.8	508.7
0.0	0.3	0.7	504.6
0.0	0.4	0.6	503.6
0.0	0.5	0.5	503.1
0.0	0.6	0.4	502.5
0.0	0.7	0.3	506.0
0.0	0.8	0.2	511.4
0.0	0.9	0.1	520.6
0.0	1.0	0.0	527.4
0.1	0.1	0.8	526.7
0.1	0.2	0.7	511.8
0.1	0.3	0.6	505.2
0.1	0.4	0.5	503.7
0.1	0.5	0.4	501.9
0.1	0.6	0.3	504.7
0.1	0.7	0.2	511.9
0.1	0.8	0.1	518.8
0.1	0.9	0.0	523.8
0.2	0.1	0.7	570.5
0.2	0.2	0.6	527.4
0.2	0.3	0.5	515.1
0.2	0.4	0.4	512.3
0.2	0.5	0.3	512.2
0.2	0.6	0.2	517.5
0.2	0.7	0.1	521.6
0.2	0.8	0.0	532.0
0.3	0.1	0.6	604.6
0.3	0.2	0.5	551.4
0.3	0.3	0.4	530.2
0.3	0.4	0.3	525.0
0.3	0.5	0.2	523.6
0.3	0.6	0.1	529.5
0.3	0.7	0.0	534.2
0.4	0.1	0.5	635.7
0.4	0.2	0.4	574.7
0.4	0.3	0.3	550.2
0.4	0.4	0.2	539.0
0.4	0.5	0.1	539.0
0.4	0.6	0.0	545.5
0.5	0.1	0.4	665.6
0.5	0.2	0.3	595.2

Table 2 continued

α	β	1-α − β	Makespans
0.5	0.3	0.2	570.2
0.5	0.4	0.1	555.5
0.5	0.5	0.0	560.5
0.6	0.1	0.3	686.5
0.6	0.2	0.2	617.0
0.6	0.3	0.1	591.1
0.6	0.4	0.0	590.7
0.7	0.1	0.2	708.2
0.7	0.2	0.1	643.8
0.7	0.3	0.0	629.2
0.8	0.1	0.1	732.6
0.8	0.2	0.0	695.5
0.9	0.1	0.0	786.1

EST+fitness approach [25]. Most previous task allocation approaches simply adopted either the best-fit principle [24] or the EFT (Earliest Finish Time) principle [2]. We showed that their performance largely depends on the workload condition and workflow property in the experiments. For example, the best-fit heuristic is effective for workflows of larger CCR when the inter-arrival time between workflows is smaller. On the other hand, our adaptive dual-criteria task group allocation method tries to balance the effects of both time gap fitness and task group's EFT. The experimental results show that our method can consistently deliver better performance than previous approaches across different workload conditions and workflow properties by adjusting the relative weights of time gap fitness and task group's EFT. In addition, the adaptive task group rearrangement mechanism in our method helps to improve the performance further especially for workflows of smaller CCR. Our adaptive dual-criteria task group allocation method achieves shorter average makespan than previous approaches across different conditions. The performance improvement ranges from 5 to 29 % for different conditions, achieving the largest performance improvement for workflows of smaller CCR. Moreover, it leads to 100 % win%, indicating that each individual workflow is benefited from our method in addition to the improved average performance.

Acknowledgments This paper is based upon work supported by National Science Council (NSC), Taiwan, under Grants no. NSC 101-2221-E-142 -002 -MY2.

References

1. Pinedo ML (2008) Scheduling: theory, algorithms, and systems. Springer, New York
2. Bittencourt LF, Madeira ERM (2008) A performance-oriented adaptive scheduler for dependent tasks on grids. Concurr Comput Pract Exp 20(9):1029–1049
3. ASKALON (2015) http://www.dps.uibk.ac.at/projects/teuta/
4. DAGman (2015) http://www.cs.wisc.edu/condor/manual/

5. Gridbus (2015) http://www.cloudbus.org/workflow/
6. Pegasus (2015) http://pegasus.isi.edu/
7. Wieczorek M, Prodan R, Hoheisel A (2008) Taxonomies of the multi-criteria grid workflow scheduling problem. In: Grid Middleware and Services, pp 237–264
8. Business Process Execution Language (BPEL) (2015) http://en.wikipedia.org/wiki/Business_Process_Execution_Language
9. Cicerre FRL, Madeira ERM, Buzato LE (2006) A hierarchical process execution support for grid computing. Concurr Comput Pract Exp 18(6):581–594
10. Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 2(13):260–247
11. Huang KC, Tsai YL, Liu HC (2015) Task ranking and allocation in list-based workflow scheduling on parallel computing platform. J Supercomput 71(1):217–240
12. Zhao H, Sakellarious R (2006) Scheduling multiple DAGs onto heterogeneous systems. In: Proceedings of the 20th international conference on parallel and distributed processing
13. Hsu CC, Huang KC, Wang FJ (2011) Online scheduling of workflow applications in grid environments. Future Gen Comput Syst 27(6):860–870
14. Hirales-Carbajal A, Tchernykh A, Yahyapour R, González-García JL, Röblitz T, Ramírez-Alcaraz JM (2012) Multiple workflow scheduling strategies with user run time estimates on a grid. J Grid Comput 10(2):325–346
15. Mu PC, Nezan JF, Raulet M (2011) A list scheduling heuristic with new node priorities and critical child technique for task scheduling with communication contention. In: Algorithm-Architecture Matching for Signal and Image Processing. Lecture Notes in Electrical Engineering, vol 73, pp 217–236
16. Garg R, Singh AK (2014) Multi-objective workflow grid scheduling using ε-fuzzy dominance sort based discrete particle swarm optimization. J Supercomput 68(2):709–732
17. Deelman E, Singh G, Kesselman C (2005) Optimizing grid-based workflow execution. J Grid Comput 3(3):201–219
18. Adabi S, Movaghar A, Rahmani AM (2014) Bi-level fuzzy based advanced reservation of Cloud workflow applications on distributed grid resources. J Supercomput 67(1):175–218
19. Bittencourt LF, Madeira ERM (2007) Fulfilling task dependence gaps for workflow scheduling on grids. In: Proceedings of the 3rd IEEE International Conference on Signal-Image Technology and Internet Based Systems (SITIS), pp 468–475
20. Yang T, Gerasoulis A (1994) DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Trans Parallel Distrib Syst 5(9):951–967
21. Bittencourt LF, Madeira ERM (2010) Towards the scheduling of multiple workflows on computational grids. J Grid Comput 8(3):419–441
22. Liou J, Palis MA (1996) An efficient clustering heuristic for scheduling DAGs on multiprocessors. In: Proceedings of the 8th symposium on parallel and distributed processing
23. Jiang HJ, Huang KC, Chang HY, Gu DS, Shih PJ (2011) Scheduling concurrent workflows in HPC cloud through exploiting schedule gaps. In: Proceedings of the 11th International Conference on Algorithms and Architectures for Parallel Processing, pp 282–293
24. Stavrinides GL, Karatza HD (2011) Scheduling multiple task graphs in heterogeneous distributed real-time systems by exploiting schedule holes with bin packing techniques. Simul Model Pract Theory 19(1):540–552
25. Tsai YL, Huang KC, Chang HY, Ko J, Wang ET, Hsu CH (2012) Scheduling multiple scientific and engineering workflows through task clustering and best-fit allocation. In: Proceedings of 2012 IEEE Eighth World Congress on Services, pp 1–8
26. Sinnen O (2007) Task scheduling for parallel systems. Wiley, New York
27. Bharathi S, Chervenak A, Deelman E, Mehta G, Su MH, Vahi K (2008) Characterization of scientific workflows. In: Proceedings of 3rd workshop on workflows in support of large-scale science (WORKS08)