Mapping Workflows on Grid Resources: Experiments with the Montage Workflow

Rizos Sakellariou and Henan Zhao and Ewa Deelman

Abstract Scientific workflows have received considerable attention in Grid computing. This paper is concerned with the issue of scheduling scientific workflows and, by considering a commonly used astronomy workflow, Montage, investigates the impact of different strategies to schedule the workflow graph. Our experiments suggest that the rather regular and symmetric nature of the Montage graph allows rather simple to implement scheduling heuristics that do not take into account the whole structure of the graph, such as Min-min, to deliver competitive performance in most cases of interest. The results support the view that sophisticated graph scheduling heuristics may not be always a prerequisite for good performance in workflow execution. Instead, mechanisms to deal with uncertainties in execution time may be of comparatively higher importance.

Key words: Grid scheduling, DAG scheduling, Montage workflow application

1 Introduction

A number of scientific applications consist of individual, standalone application components, each often independently designed and developed, which are then combined in pre-defined ways to perform large-scale scientific analysis. In recent years, *scientific workflows* [9, 22] have been used to refer to the process of bringing the individual components together and specifying their interactions in a systematic

Rizos Sakellariou
School of Computer Science, University of Manchester, Manchester M13 9PL, United Kingdom, e-mail: rizos@cs.man.ac.uk

Henan Zhao
School of Computer Science, University of Manchester, Manchester M13 9PL, United Kingdom

Ewa Deelman
USC Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA90292, USA

way. Once a scientific workflow (or simply workflow) has been assembled, a key problem that needs to be addressed relates to *mapping* the components of the workflow onto distributed resources, that is, what node of the graph is going to execute on what resource. This problem needs to take into account all constraints (for instance, some components may have to execute on specific nodes) as well as to optimize for various objectives such as the overall completion time of the workflow, resource usage, (monetary) cost of using the resources, etc.

Since most known types of workflows appear to be typically represented by a Directed Acyclic Graph (DAG), there has been a considerable amount of work trying to solve this workflow mapping problem using DAG scheduling heuristics [4, 25, 27]. Such heuristics generally aim at minimizing the cost of running the critical path of the graph. However, it has been argued [12] that such heuristics, although worthwhile, might not be substantially more efficient in the particular context of workflow scheduling on the grid; their benefits might be outweighed by their additional complexity. This argument can be reinforced by the easy to make observation that DAGs representing many real-world workflow applications seem to have a somewhat regular and symmetric structure. As a consequence, simple scheduling approaches, which do not consider the whole structure of the DAG at once, might be a good alternative for workflow scheduling. Following the broad classification in [4, 26], we term the latter approaches as *local* (or task-based), since their decision making strategy relies on locally optimal choices, as opposed to *global* (or *workflow-based* according to [4]) strategies that consider the whole structure of the graph.

To the best of our knowledge, there has been only limited work trying to evaluate and quantify the advantages and disadvantages of local strategies versus global strategies when mapping workflows on the grid. In [4], it has been found that the difference in the makespan between a global and a local strategy using the well-known Montage workflow [2, 17] was less than 0.3%. However, the difference would increase to more than 100% for data-intensive workflows, where the communication cost would dominate computation (see Table 1 in [4]). In contrast, the experimental study in [12] has indicated that a simple local strategy can also cope with data-intensive cases and high communication to computation cost; however, the authors of this study notice limitations for the local strategy in the case of sparse DAGs, which are due to the small degree of parallelism or the small number of dependencies amongst the tasks.

In this paper, we contribute to the quantitative evaluation of the advantages and disadvantages of local versus global strategies by considering the impact of *uncertainty* in workflow mapping, using, in our study, a workflow that implements a widely mentioned astronomy application to build mosaics of the sky, Montage [2, 17]. Since the initial mapping decisions for the workflow are made on the basis of static estimates, a key factor in the evaluation of the performance of local vs global strategies is how well the initial mapping onto resources performs when there are deviations from the estimated execution time of each task.[1] The ability of

[1] Such deviations, from the initially estimated execution time, may be due to any reason: wrong prediction, resource load, etc. In principle, these deviations can be corrected at run-time using, for example, rescheduling [21] or adaptive [14] techniques. However, there is also a need to minimize

a scheduling algorithm to produce a schedule that is affected as little as possible by run-time changes is known as *robustness* and, for limited degrees of uncertainty, has been studied elsewhere [7]. In this paper, we use large degrees of uncertainty, which include actual execution times that may be up to 4 times higher than initially estimated (these times can also be shorter than the estimates). Our simulation results validate our hypothesis: when using Montage, variations between local and global workflow mapping strategies are insignificant (at most about 3.5%) and appear to be consistent regardless of the degree of uncertainty with respect to the initial execution times estimates. Instead, by using a tweaked version of the Montage DAG, with a smaller number of edges, and longer parallel paths in the graph, the variation in execution time between local and global workflow mapping strategies becomes more profound, up to about 12%. This indicates some correlation of the performance of these mapping strategies strategies with the type of the DAG they are applied to.

The remainder of the paper is structured as follows. Section 2 provides some background on the problem of DAG/workflow scheduling and relevant heuristics. Section 3 gives a motivating example that highlights the possible differences in performance that some heuristics may exhibit depending on the structure of the graph considered. Section 4 is trying to assess the possible differences in performance between a local, task-based, approach and a global, workflow-based, approach when scheduling Montage [10], a commonly cited application used in astronomy to create a large mosaic image of the sky from many smaller astronomical images. Finally, Section 5 concludes the paper.

2 Background

The model used for the representation of a workflow is a Directed Acyclic Graph (DAG), where nodes (or tasks) represent computation and edges represent data or control flow dependences between nodes. A set of machines is assumed to be already available and known. These machines and the network links between them are heterogeneous: tasks may need a different amount of time to execute on each machine and the transmission of data between different machines is not the same. A machine can execute only one task at a time, and a task cannot start execution until all data from its parent nodes is available. The scheduling problem is to assign the tasks onto machines so that precedence constraints are respected and the makespan is minimized.

In order to be able to make sensible scheduling decisions, it is assumed that information about the estimated execution time of each task on each machine is available. In addition, it is assumed that there are estimates about the speed of the links connecting the machines available. This information, used in conjunction with the amount of data that may need to be transferred before a task starts its execution,

---

the overhead associated with rescheduling and/or adaptivity and keep the number of times when such an action occurs small. Our work focuses on how the initial workflow mapping decisions can help in this respect.

can provide an estimate about the earliest possible start time of a task whose parents have finished their execution.

The problem of scheduling DAGs onto parallel resources is well studied in the literature [13]. In recent years, partly as a result of the emergence of Grid systems and applications such as workflows, additional research has focused on DAG scheduling algorithms for heterogeneous systems [20, 23, 27], as well as their performance in the context of the uncertainties typically associated with the actual execution time of tasks [7, 11, 15, 21]. A growing amount of work has also evaluated DAG scheduling algorithms in the context of specific workflow applications [4, 16, 24].

As already mentioned, it is common to classify DAG scheduling algorithms according to whether scheduling decisions are made *locally*, with reference to just a task or a set of tasks, or *globally*, with reference to the whole DAG (workflow) [4, 26]. A commonly used heuristic in the former class (task-based) is *Min-min*, originally developed in the context of scheduling independent tasks [5]. This can also be applied in the context of scheduling DAGs, since, at any point in time, the tasks that are considered to be eligible for scheduling are, by definition, independent. This is because the eligible tasks are tasks whose data is available, hence, their parents have finished execution. The key idea of Min-min is to find, for each eligible task, the machine that gives the earliest completion time for this task. Then, the task with the minimum earliest completion time is chosen for scheduling. The process is then repeated with the remaining task as well as any new tasks that become eligible (as a result of the completion of their parents). As noticed in [4], "the intuition behind this heuristic is that the makespan increases the least at each iterative step, hopefully resulting in a small makespan for the whole workflow".

A commonly cited global heuristic in the context of DAG scheduling for heterogeneous systems is HEFT [23]. HEFT first orders tasks by assigning a value to each task. This value roughly corresponds to the cost to reach the exit node from this task. Then, tasks are scheduled using this order to the machine that gives the earliest completion time. The key idea (of this *list scheduling* based heuristic [18]) is to give higher priority to tasks on the critical path. Several variations to assign weights and prioritize the tasks have been studied in [28]. Following observations about the impact of such variations, HBMCT [20] tries to improve the performance of HEFT by relaxing the requirement to schedule tasks in a strict order of their ranking, considering groups of independent tasks. Among the global heuristics, it is also worth mentioning the workflow-based allocation algorithm (WBA) [4], which compares several alternative workflow schedules before the final schedule is chosen, based on a generalized greedy randomized adaptive search procedure.

Typically, heuristics that make decisions locally (task-based) are simpler and faster, whereas heuristics that make decisions globally, with reference to the whole workflow (workflow-based), have the potential to produce a shorter makespan at the expense of increased complexity. Such a potentially shorter makespan is a consequence of their ability to consider the whole graph at once and, hence, give appropriate priority in execution to tasks in the critical path. The hypothesis considered in this paper, however, is that the regular and symmetric structure of scientific work-

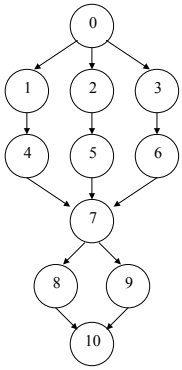
flows does not stand to benefit from global heuristics. To illustrate the impact that the structure of the graph may have, we present two examples in the next section.

3 A Motivating Example

The simplicity of local, task-based approaches as opposed to the shorter makespan expected to be produced by a global, workflow-based approach is the key trade-off to assess when selecting the heuristic that would be more suitable to schedule a certain application. In the context of arbitrary DAGs, global heuristics, which are capable of tracking the critical path, are expected to give better performance. However, it is questionable whether the examples of scientific workflows that exist can be regarded as *arbitrary* DAGs. Instead, all the evidence available seems to suggest that many scientific workflows have a regular and rather symmetric structure. Many appear to consist of sequences of fan-out (where the output of a task is input to several children) and fan-in (where the output of several tasks is aggregated by a child). Typically, the outcome of fan-out procedures is identical tasks that simply operate on different data (indicating the exploitation of data parallelism). For examples, we refer to workflows such as Montage (see Figure 9 in [10]), Chimera (see Figures 6 and 7 in [1]), LIGO (see Figure 4 in [19]), WIEN2k (see Figure 6 in [24]), Invmod (see Figure 7 in [24]) and AIRSN (see Figure 5 in [29]) as well as the workflows studied in [3].

In order to illustrate the possible differences in the schedule resulting from a local heuristic, Min-min, and a global heuristic, HBMCT, and how they are affected by the structure of the graph, consider the examples in Figures 1 and 2. The graph in Figure 1 has a rather regular, symmetric structure. We view it as regular because it consists of a sequence of fan-out and fan-in (repeated twice) and symmetric because the independent subgraphs created during the fan-out procedure are identical (in terms of their structure). In addition, the execution time of the tasks on three different machines, M0, M1, M2, is similar, although not always identical (see the table at the top right of the figure), while the cost of sending data from one task to another (when these tasks are executed on different machines) is set to 8 time units. The schedule produced by Min-min and HBMCT is shown at the bottom of the figure (Min-min is on the left-hand side). Both heuristics produce a schedule of an identical length (130 time units), although task assignments to machines are different (and tasks are not necessarily assigned by HBMCT to the fastest machine for the task).

Conversely, the example in Figure 2 considers a graph with an asymmetric structure (for example, each of nodes 6, 7, and 8, which are at the same layer, has a different number of parents: 1, 2, 3, respectively). Also, the execution time of each task on three different machines shows a higher degree of heterogeneity, while the time needed to send data between different machines varies, with the link between machines M0 and M2 being the slowest. In this case, the makespan of the schedule produced by Min-min (left-hand side at the bottom of the figure) is 143.6 time



task	M0	M1	M2
0	25	27	24
1	24	24	24
2	25	25	26
3	24	24	25
4	16	16	16
5	16	15	16
6	14	14	14
7	10	11	11
8	12	12	12
9	11	11	11
10	15	17	16

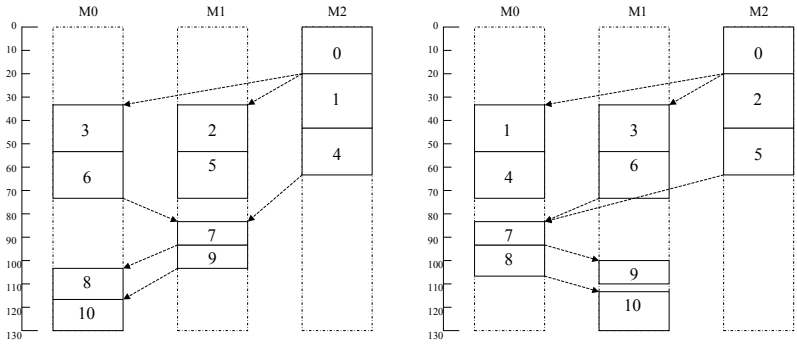
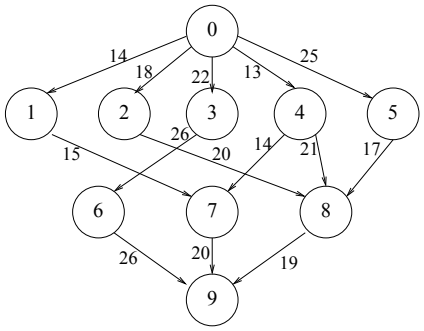


Fig. 1 Scheduling a regular and symmetric graph using Min-min and HBMCT.

units, while the makespan of the schedule produced by HBMCT (right-hand side) is 124.6, which is approximately 13% better than Min-min. This is because a global approach, such as HBMCT, is capable of giving priority to tasks of those (critical) paths in the graph that have a higher cost (such as the paths consisting of the tasks 0, 1, 7, and 9 and 0, 5, 8, and 9) as opposed to tasks of other paths.

The question that this paper is set to investigate is whether a typical scientific workflow, such as Montage, would stand to benefit significantly from a global, workflow-based approach for scheduling. The hypothesis is that the regular and symmetric structure of the graph in workflows, such as Montage, is the key factor



task	M0	M1	M2
0	17	19	21
1	22	27	23
2	15	15	9
3	4	8	9
4	17	14	20
5	30	27	18
6	17	16	15
7	49	49	46
8	25	22	16
9	23	27	19

machines	time for a data unit
M0 - M1	0.9
M1 - M2	1.0
M0 - M2	1.4

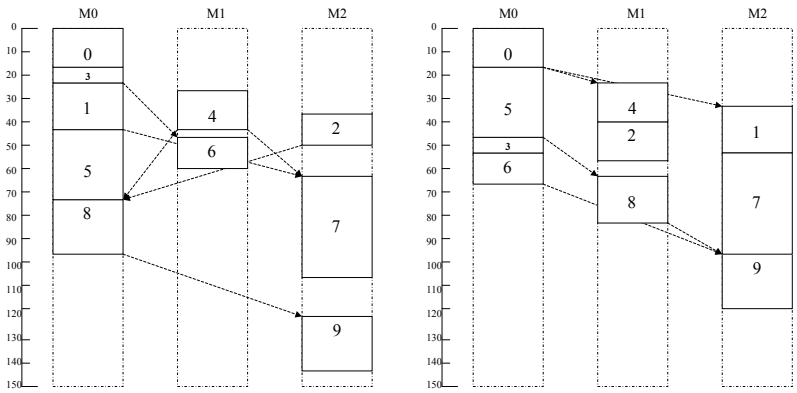


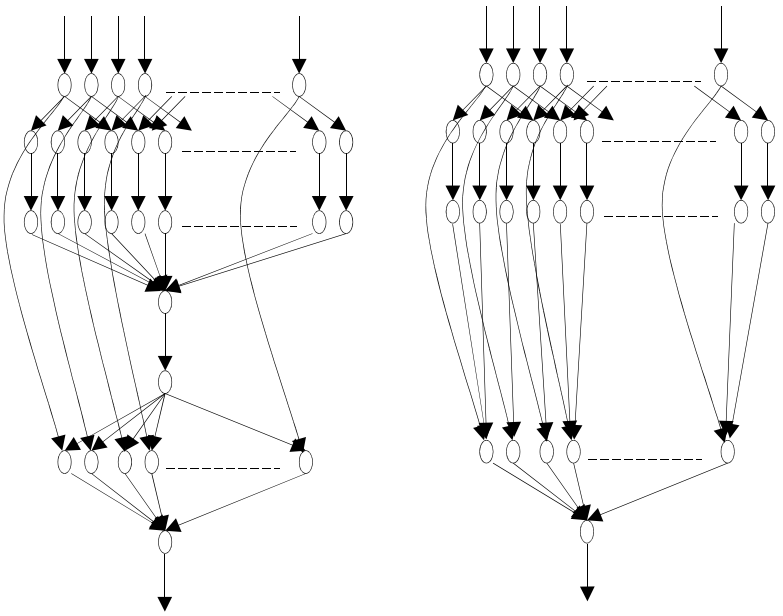
Fig. 2 Scheduling an asymmetric graph using Min-min and HBMCT.

that makes the performance of local approaches for scheduling equally competitive to the performance of the more complex global strategies.

4 Experimental Evaluation

4.1 The Simulator and Settings

For the purposes of our evaluation we used a grid simulator built on the top of the network simulator NS-2 [6], which was also used in earlier research [4]. We only briefly describe the grid simulator here; more details can be found in [4]. The



(a) 57-task Montage workflow

(b) 55-task tweaked Montage (without the two middle jobs)

Fig. 3 Graphs of the workflows used in the experiments (based on Montage [10]).

simulator models resources, networks connecting different resources, jobs and files. Each file is considered as a separate object. Each resource (i.e., a site) consists of several hosts and each host can run jobs and store files. The simulator can be adapted to include different scheduling algorithms to allocate jobs to sites; as part of this work we adapted it to include HBMCT [20]. Thus, the simulator includes three different approaches for scheduling: one local, task-based, Min-min, and two global, workflow-based, from which one is based on a list scheduling principle, HBMCT, whereas the other, WBA, is not.

The workflow application we considered is Montage [2, 10, 17]. The instance of the workflow we used is shown in Figure 3(a). This contains 57 tasks (adding the tasks of each level from the top the sum is 13+14+14+1+1+13+1). The original version of Montage assumes that tasks at the same level have a similar computation cost. In some of the experiments we considered different types of variation in the execution time of the tasks at each level. In addition, in order to experiment with the structure of the graph, a variant of the Montage graph has also been used; this is shown in Figure 3(b). The latter graph does not contain the middle two tasks of the original graph, thus reducing the degree of synchronization needed and increasing the relative importance of the independent (i.e., parallel) paths in the graph when it comes to the overall makespan.

QoI	57-task Montage				55-task tweaked Montage			
(%)	Min-min	WBA	HBMCT	%	Min-min	WBA	HBMCT	%
0	7802.6	7785.4	7796.9	0.22	7302.1	7288.5	7234.9	0.92
50	9146.0	9060.4	9097.7	0.93	9750.3	9710.6	9688.0	0.64
100	9949.4	9878.8	9930.2	0.71	10293.4	9696.1	9627.7	6.47
200	12562.7	12436.4	12420.3	1.13	11949.0	11432.3	11370.5	4.84
300	12763.4	12531.3	12305.2	3.58	11273.3	10410.1	10392.1	7.82
400	13125.0	12710.7	12687.3	3.33	11287.0	10551.1	10523.8	6.76

Table 1 Overall execution time when tasks at each level have a similar (estimated) execution time.

In order to capture the degree of variation expected between the estimated execution time and the actual execution time of each task, we adopted the notion of the *Quality of Information* (QoI) [8, 21]. This corresponds to an upper bound on the percentage of error that the static estimate may have with respect to the actual execution time. So, for example, a percentage error of 50% indicates that the actual (run-time) execution time of a task will be within 50% (plus or minus) of the static estimate for this task. This value is always positive. In our experiments, we consider values for QoI of 0% (perfect estimates), 50%, 100%, 200%, 300%, and 400%.

4.2 Results and Discussion

The objective of our experiments has been to quantify and assess the difference in the performance of the schedule produced by Min-min, HBMCT and WBA, and test our hypothesis that the symmetric and regular structure of the graph does not have much to gain from global scheduling strategies. To achieve our objective, we used:

I Five different values for QoI to check run-time deviations, which are up to four times the estimated execution times of tasks.

II Two different graph structures, one corresponding to a 57-task instance of the original Montage workflow, shown in Figure 3(a), and one corresponding to a 55-task tweaked version of Montage, shown in Figure 3(b), which reduces the degree of synchronization by removing two tasks.

III Three different assumptions about the *estimated* execution time of tasks at each level of the graph. These assumptions are: (a) tasks at each level have a similar execution time; (b) all tasks except the rightmost task at each level have an execution time which is twice the execution time of the rightmost task; and (c) the leftmost task at each level has an execution time which is ten times the execution time of the remaining tasks at the same level.

In the following, we group the results using the three different assumptions for the execution time of the tasks at each level. We also note than, in all cases, the results are averaged over 20 runs and there are always 6 machines available to schedule the tasks of the workflow.

QoI	57-task Montage				55-task tweaked Montage			
(%)	Min-min	WBA	HBMCT	%	Min-min	WBA	HBMCT	%
0	14437.2	14109.4	14120.8	2.27	14667.2	14059.0	13527.3	7.77
50	16024.3	15327.1	15545.2	4.35	15994.0	15197.2	15068.9	5.78
100	18774.0	18258.0	18223.5	2.93	17466.2	15855.5	15356.1	12.08
200	33625.5	32485.0	32448.7	3.50	28843.6	27598.7	26832.8	6.97
300	39438.9	37937.1	38066.2	3.81	39151.8	37430.0	36557.0	6.63
400	51123.0	50701.0	50278.6	1.65	50706.6	49861.0	49211.3	2.95

Table 2 Overall execution time when all tasks at each level except the rightmost task have an (estimated) execution time which is twice the (estimated) execution time of the rightmost task.

4.2.1 All tasks at each level have a similar (estimated) execution time

Table 1 shows the overall execution time (as provided by the simulator) of the two different variants of Montage considered, when tasks at each level have a similar execution time. The leftmost column shows the upper bound of Quality of Information (QoI) considered in each row. The next three columns show the execution time for each scheduling algorithm used with the 57-task Montage. The fourth column shows the percentage gain of the best global scheduling strategy (that is, the best of WBA and HBMCT) as compared to the task-based Min-min (this value is computed as $100 \times (1 - \min(m_{WBA}, m_{HBMCT})/m_{MinMin})$), where m is the makespan of the corresponding strategy. From the results, it can be seen that in the case of the 57-task Montage, Min-min has a performance which is comparable to the performance of the workflow-based heuristics. Thus, the gain of global, workflow-based heuristics is less than 1% if the value of QoI is up to 100%, reaching a maximum of just 3.58% in more extreme cases where the value of QoI is higher. The comparatively worse performance of Min-min in the case of high variations in the actual execution time (as opposed to the estimated execution time) may be due to its lower robustness (comparing to HBMCT and WBA), a result also corroborated from [7]. The gain is higher in the case of the tweaked 55-task Montage, due to the higher degree of parallelism, which opens up more opportunities for different schedules. As an aside remark, it is noted that, with the 57-task Montage, WBA has a slightly better performance than HBMCT for small values of QoI, whereas HBMCT tends to perform better as the value of QoI increases (a result also corroborated from [7]). The performance of HBMCT is consistently better than WBA (although not by much) in the case of the 55-task tweaked Montage.

4.2.2 All tasks at each level, except the rightmost task, have an (estimated) execution time which is twice the (estimated) execution time of the rightmost task

Table 2 shows the overall execution time when there is some variation in the execution time of the tasks at each level, in particular when the rightmost task has

QoI	57-task Montage				55-task tweaked Montage			
(%)	Min-min	WBA	HBMCT	%	Min-min	WBA	HBMCT	%
0	8213.0	8190.6	8204.7	0.27	7685.6	7674.8	7662.0	0.31
50	9297.3	9286.6	9276.3	0.23	9311.3	9199.4	9034.7	2.97
100	10105.2	10071.0	10151.0	0.34	10323.7	10002.8	9839.0	4.70
200	12946.5	12836.4	12633.9	2.41	14181.6	12684.3	12557.3	11.45
300	14277.8	14001.6	14247.0	1.93	13926.9	12578.5	12482.6	10.37
400	21896.4	21719.1	22035.1	0.81	23462.0	21214.3	21092.0	10.10

Table 3 Overall execution time when the leftmost task at each level has an (estimated) execution time which is 10 times the (estimated) execution time of the other tasks at the same level.

an execution time which is half the execution time of the other tasks at the same level. The impact of this variation is that it creates a particular shorter path as opposed to several longer paths in the graph. The Min-min heuristic still manages to produce reasonably efficient schedules comparing to WBA and HBMCT. Even though these schedules are not as efficient as before, their performance is still only up to 4.35% worse than the performance of the more sophisticated, workflow-based heuristics. We observe the same pattern of behaviour as before; the efficiency of Min-min is worse in the case of the 55-task tweaked Montage. Furthermore, we observe the same pattern of behaviour when we compare the performance of WBA and HBMCT.

4.2.3 The leftmost task at each level has an (estimated) execution time, which is 10 times the (estimated) execution time of the other tasks at the same level

Table 3 shows the overall execution time of the workflows when the variation in task execution time is due to an increase in the execution time of the leftmost task at each level, which is 10 times the execution time of the other tasks at the same level. The impact of this variation in execution time is that it creates one relatively long critical path in the graph. It can be seen that in the case of the 57-task Montage the performance of Min-min is clearly comparable to the performance of WBA and HBMCT; it is at most 2.41% worse. In one case (for a value of QoI equal to 400), Min-min even outperforms HBMCT. Same as before, the performance of Min-min worsens in the case of the 55-task tweaked Montage. In fact, interestingly enough, in this case, the deficiency of Min-min is worst comparing to the two previous sets of results.

4.2.4 Summary

The results in Tables 1, 2, 3 support our view that in the case of the regular and symmetric Montage application, a local strategy, such as Min-min, does not appear

to be inferior to sophisticated global strategies, both when there are large run-time deviations from the actual estimated execution times (i.e., large values of QoI) and when there are differences in the estimated execution time between tasks at each level (as it is the case in Tables 2 and 3). As already hypothesized, this property appears to be a consequence of the structure of the graph. This is supported by our experiments with the tweaked Montage, where Min-min clearly lags behind as a result of its inability to handle efficiently the critical paths. Finally, it is interesting to observe that the introduction of uncertainty in the execution time (which may also cause individual tasks to run faster) on average increases the overall execution time of the workflow by up to more than three times with respect to the statically estimated execution time. This observation suggests that, regardless of the quality of the scheduling heuristic that is used to find an initial mapping for tasks of a DAG, there may be, comparatively, much more to be gained in performance by run-time rescheduling, a finding also supported by the experiments in [14].

5 Conclusion

This paper has compared the performance of the schedule produced by three different approaches for scheduling workflows on the Grid. It has been found that scheduling a workflow, such as (a 57-task) Montage, using a simple heuristic that makes only local decisions, Min-min, results in performance which is comparable to the performance obtained with more sophisticated, workflow-based scheduling heuristics such as WBA or HBMCT. The performance of Min-min worsens slightly, but not significantly, when high variations between the estimated and the actual execution time of the tasks of the workflow exist. Min-min also exhibits good performance if the workflow consists of a single critical path with significantly longer execution time. The performance of Min-min drops when there is a single path of shorter length and multiple critical paths in the workflow (cf. Table 2). Its performance worsens even further in the case of a workflow with multiple long parallel paths (cf. the results with the 55-task tweaked Montage). It is noted that this may not be a common case in practice, since, as already mentioned, the structure of several commonly cited workflows consists of alternating fan-out and fan-in phases.

In summary, the results are encouraging in that they suggest that Min-min may be sufficiently efficient in the context of scheduling certain classes of scientific workflows on the Grid even when there are uncertainties in the estimated task execution times. It is noted that the small performance deficits of Min-min can be offset by its inherent simplicity and the ease with which it can be adopted to perform adaptive rescheduling at run-time, options that have been shown to be relatively more important when addressing run-time changes [14, 21]. Future work can expand the preliminary experimental study of this paper. Also, it can try to find out under what circumstances the relative importance of run-time changes dominates the choice of an appropriate heuristic for the initial mapping onto the resources.

References

1. J. Annis, Y. Zhao, J. Voeckler, M. Wilde, S. Kent, and I. Foster. Applying Chimera virtual data concepts to cluster finding in the Sloan Sky Survey. In: *Supercomputing'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, Los Alamitos, CA, USA, IEEE Computer Society Press, 2002, pp. 1-14.
2. G. B. Berriman, J. C. Good, A. C. Laity, *et al*. Montage: A Grid Enabled Image Mosaic Service for the National Virtual Observatory. In *Astronomical Data Analysis Software & Systems (ADASS) XIII*, 2003.
3. S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi. Characterization of Scientific Workflows. In *Proceedings of the 3rd Workshop on Workflows in Support of Large-Scale Science* (WORKS 2008), November 2008.
4. J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy. Task Scheduling Strategies for Workflow-based Applications in Grids. In *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, 2005.
5. T. D. Braun, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, vol. 61, pp. 810-837, 2001.
6. L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *Computer*, vol.33(5), pp.59-67, May 2000. See also http://www.isi.edu/nsnam/ns.
7. L.-C. Canon, E. Jeannot, R. Sakellariou, and W. Zheng. Comparative evaluation of the Robustness of DAG Scheduling Heuristics. In *Grid Computing: Achievements and Prospects* (eds: S. Gorlatch, P. Fragopoulou, T. Priol), Springer, 2008, pp. 73-84. An extended version is available as CoreGRID Technical Report TR-0120, December 2007.
8. H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Heuristics for scheduling parameter sweep applications in Grid environments. In *Proceedings of the 9th Heterogeneous Computing Workshop (HCW'00)*, 2000, pp. 349-363.
9. 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*, 25, 2009, pp. 528-540.
10. E. Deelman, G. Singh, M. H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programning*, 13(3), 2005, pp. 219-237.
11. F. Dong and S. G. Akl. PFAS: A Resource-Performance-Fluctuation-Aware Workflow Scheduling Algorithm for Grid Computing. In *Proceedings of IPDPS 2007*, 2007.
12. R. Huang, H. Casanova, and A. A. Chien. Using Virtual Grids to Simplify Application Scheduling. In *Proceedings of IPDPS 2006*, 2006.
13. Y.-K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31(4), December 1999, pp. 406-471.
14. K. Lee, N. W. Paton, R. Sakellariou, E. Deelman, A. A. A. Fernandes, and G. Mehta. Adaptive Workflow Processing and Execution in Pegasus. In *3rd International Workshop on Workflow Management and Applications in Grid Environments* (WaGe08) (in *Proceedings of the Third International Conference on Grid and Pervasive Computing Symposia/Workshops*, May 25-28 2008, Kunming, China), 2008, pp. 99-106.
15. M. M. Lopez, E. Heymann, and M. A. Senar. Analysis of Dynamic Heuristics for Workflow Scheduling on Grid Systems. In *Proceedings of the 5th International Symposium on Parallel and Distributed Computing* (ISPDC), 2006, pp. 199-207.
16. A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, and L. Johnsson. Scheduling Strategies for Mapping Application Workflows onto the Grid. In *IEEE International Symposium on High Performance Distributed Computing (HPDC 2005)*, 2005.
17. Montage. An Astronomical Image Mosaic Engine. http://montage.ipac.caltech.edu/

18. M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Springer, 2008.
19. A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi. Scheduling Data-Intensive Workflows onto Storage-Constrained Distributed Resources. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid* (CCGrid'07), 2007, pp. 401-409.
20. R. Sakellariou and H. Zhao. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. In *Proceedings of the 13th Heterogeneous Computing Workshop (HCW'04)*, Santa Fe, New Mexico, USA, on April 26, 2004.
21. R. Sakellariou and H. Zhao. A low-cost rescheduling policy for efficient mapping of workflows on grid systems. *Scientific Programming*, 12(4), December 2004.
22. I. J. Taylor, E. Deelman, D. B. Gannon, and M. Schields. *Workflows for e-Science. Scientific Workflows for Grids*. Springer, 2007.
23. H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3), March 2002, pp. 260-274.
24. M. Wieczorek, R. Prodan, and T. Fahringer. Scheduling of Scientific Workflows in the ASKALON Grid Environment. In *SIGMOD Record*, volume 34(3), September 2005.
25. M. Wieczorek, R. Prodan, and T. Fahringer. Comparison of Workflow Scheduling Strategies on the Grid. In *Proceedings of the Second Grid Resource Management Workshop* (GRMW'2005), Springer, LNCS 3911, 2006, pp. 792-800.
26. J. Yu and R. Buyya. A Taxonomy of Scientific Workflow Systems for Grid Computing. In *SIGMOD Record*, volume 34(3), September 2005.
27. J. Yu, R. Buyya, and K. Ramamohanarao. Workflow Scheduling Algorithms for Grid Computing. In *Studies in Computational Intelligence*, volume 146, Springer, 2008, pp. 173-214.
28. H. Zhao and R. Sakellariou. An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm. In *Euro-Par 2003*, Springer-Verlag, LNCS 2790, 2003.
29. Y. Zhao, J. Dobson, I. Foster, L. Moreau, and M. Wilde. A notation and system for expressing and executing cleanly typed workflows on messy scientific data. In *SIGMOD Record*, volume 34(3), September 2005, pp. 37-43.