Dynamic Resources Configuration for Coevolutionary Scheduling of Scientific Workflows in Cloud Environment

Alexander A. Visheratin(✉), Mikhail Melnik, and Denis Nasonov

ITMO University, Saint Petersburg, Russia
alexvish91@gmail.com, mihail.melnik.ifmo@gmail.com, denis.nasonov@gmail.com

Abstract. Modern composite scientific applications, also called scientific workflows, require large processing capacities. Cloud environments provide high performance and flexible infrastructure, which can be easily employed for workflows execution. Since cloud resources are paid in the most cases, there is a need to utilize these resources with maximal efficiency. In this paper we propose dynamic resources coevolutionary genetic algorithm, which extends previously developed coevolutionary genetic algorithm for dynamic cloud environment by changing computational capacities of execution nodes on runtime. This method along with using two types of chromosomes – mapping of tasks on resources and resources configuration – allows to greatly extend the search space of the algorithm. Experimental results demonstrate that developed algorithm is able to generate solutions better than other scheduling algorithms for a variety of scientific workflows.

Keywords: Workflow scheduling · Dynamic coevolution · Coevolutionary algorithm · Genetic algorithm · Virtualization

1 Introduction

Composite scientific applications containing thousands tasks today drive scientific progress in many areas. For example, CyberShake application, which is used by the Southern California Earthquake Center (SCEC) to characterize earthquake hazards, consists of 815823 execution tasks, reads 217 terabytes of data and writes 920 gigabytes [1]. Workflow formalism is used for organization of execution of such large applications in the distributed high-performance environments within Workflow Management Systems (WMS) [2, 3]. The main feature of WMS is a workflow scheduling, which allows to efficiently utilize resources of the environment, minimize execution time and cost of resources for the paid platforms. Workflow scheduling also allows to meet various constraints, like execution deadlines or limited budget [4, 5].

Workflow scheduling is an NP-complete problem [6] and there are two main classes of algorithms used to solve it. The first class is heuristic algorithms, which provide determined solution in a short time. Heterogeneous Earliest Finish Time (HEFT) algorithm [7] is very popular algorithm of this group – it provides very efficient solutions in

H. Pérez García et al. (eds.), *International Joint Conference SOCO'17-CISIS'17-ICEUTE'17 León, Spain, September 6–8, 2017, Proceedings,* Advances in Intelligent Systems and Computing 649,
DOI 10.1007/978-3-319-67180-2_2

a very short time. The second class of algorithms for the workflow scheduling is metaheuristic algorithms. These algorithms usually provide better solutions, than heuristics, but require much more time for their execution. Among popular metaheuristic algorithms are genetic algorithm, ant colony optimization, particle swarm optimization, GRASP [8], memetic algorithm [9]. Better quality of resulting solutions is achieved by searching through a solution space instead of using predefined algorithm.

In this work we present a Dynamic Resources Coevolutionary Genetic Algorithm (DRCGA), which uses two types of chromosomes – resources configurations and mappings of tasks on resources – to extend a search space in order to generate better solutions than standard GA with one population. The main improvement of the developed algorithm in comparison with the scheme proposed earlier [10] is ability of the algorithm to schedule changes in the configuration of virtual machines on runtime to adapt the environment to the changing requirements of the workload.

2 Related Works

Liu et al. [11] proposed the coevolutionary genetic algorithm for deadline-constrained scientific workflow scheduling problem in clouds. The main idea of the algorithm is in the existence of two coevolution populations. The first population is composed of several sub-populations which evolve the original scheduling problem – the assignment of workflow's tasks on various computation nodes. Simultaneously, the second population evolves mutation and crossover probabilities for each sub-population. Therefore, the second population allows to adapt these parameters during the evolution process. Moreover, authors proposed an adaptive penalty function for the estimation of penalty in fitness function. This adaptive penalty function depends on the proportion of feasible individuals in the population. Authors compared their proposed CGA with several scheduling algorithms, including random, heuristic HEFT and metaheuristic PSO and GA. Experiments were provided in the simulated environment under various workflows, such as Epigenomics, Montage, Inspiral, CyberShake. The results of experiments showed, that all metaheuristic algorithms together with CGA were able to find a solution, which doesn't exceed a deadline. However, the main optimization criterion was the execution cost, and the proposed CGA outperformed other algorithms almost in all cases.

The heuristic list-based scheduling algorithm called Improved Predicted Earliest Finish Time (IPEFT) was introduced by Zhou et al. [12]. The algorithm is based on the computation of optimistic cost table which is used to identify priorities of tasks and the computation of critical node cost table, which is used for selection of an optimal node for a task. This work is a continuation of authors' previous work, where original PEFT algorithm was proposed. Since, proposed IPEFT is a heuristic, the main goals of IPEFT algorithm is to outperform other low-complexity heuristic algorithms (HEFT, PEFT) and save the time-complexity. According to the results of experiments, proposed IPEFT outperforms HEFT and PEFT algorithms with the same quadratic complexity on randomly generated and several real-world workflows.

The Global League Championship Algorithm (GBLCA) for scheduling of scientific applications was proposed in [13]. This algorithm is one of population based metaheuristics as GA, which is inspired from sport league championships. The experiments were conducted in CloudSim simulation environment. The input data of incoming workload for the experiments is taken from Parallel Workload Archive in SWF format, generated by San Diego Supercomputer Center. The proposed algorithm was compared with GA and ACO. As a performance metrics, authors used makespan and response time. The results of the experiments showed the efficiency of proposed GBLCA in comparison to GA and ACO by makespan and response time for various number of tasks in the input workload.

The energy-efficient online scheduling algorithm (EONS) is proposed by Chen et al. [14]. As a list-based heuristic algorithm, it includes two phases: tasks ranking and node selection. The ranking phase is based on the estimation of latest start time of workflows' tasks. The main idea of selection phase is in the ability to rescale virtual machines up and down. Moreover, algorithm may start a new inactive physical host, if it is required to place a new VM. In addition, authors provided a complete problem definition with corresponding workflow, system, performance and energy consumption models. Therefore, proposed EONS algorithm with defined models allows to outperform several popular heuristic algorithms in terms of energy consumption, resource utilization and do not exceed the specified deadlines.

3 Problem Statement

In order to define a formal problem statement for the workflow scheduling, we need to specify representations for both workload and execution environment. Workload consists of several workflows and can be defined as follows:

$$W = \{w_i\} = \{< A_i, E_i >\}$$

where every workflow w_i contains a set of tasks $A_i = \left\{a_j^i\right\}$ and a set of dependencies between tasks:

$$E_i = \left\{e_{j,k}^i\right\} = \left\{< \left(a_j^i, a_k^i\right), d_{j,k}^i >\right\}$$

where every entry is a tuple of parent (a_j^i) and child (a_k^i) tasks, and a volume of data $d_{j,k}^i$, which is transferred from a_j^i to a_k^i.

To simplify the representation of the workload we can use global indexing of tasks. In this case all workflows are combined into one, which contains all tasks and dependencies between them:

$$W = \{< \{a_i\}, \{e_{j,k}\} >\}$$

Execution environment is defined using two sets. The first one – R – is a set of physical resources, each of which includes hardware characteristics, such as CPU cores number, CPU frequency, RAM volume, etc.:

$$R = \{r_i | (cpu_i, ram_i, gpu_i, \ldots) \in r_i\}$$

The second set – $\overline{B}$ – represents a set of network routes between resources:

$$\overline{B} = \left\{ \overline{(b_l)} | \{r_k\} \in \overline{(b_l)} \right\}$$

Every resource is split into several virtual resources. All virtual resources form a set of computational nodes N, which can be used for scheduling tasks on them. Computational node inherits some characteristics of the parent resource:

$$N = \{n_i | n_i \in r_k; (cpu_i, \ldots) \le (cpu_k, \ldots)\}$$

Now we can formulate the scheduling problem. Schedule S is a mapping of tasks onto resources:

$$S = \{(a_i, n_k)\}$$

Performance P of the task a_i on the node n_k is obtained using performance models, which can be generated analytically or statistically:

$$P = p(a_i, n_k)$$

There can be several optimization criteria $G = \{g_i(S)\}$, which we need to maximize, and several constraints $C = \{c_i(S)\}$. All solutions must meet the constraints, otherwise they can be marked as invalid or get a high penalty, depending on the type of the constraint.

Taking into account all aforementioned, we can formulate a problem statement. The main aim of scheduling is to find an optimal schedule S_{opt}:

$$\exists S_{opt} : \forall S' \ne S_{opt} : \begin{cases} g_i(S_{opt}) \ge g_i(S'), \forall g_i \in G \\ c_j(S_{opt}) \le c_j(S'), \forall c_j \in C \end{cases}$$

In other words, an optimal schedule maximizes the optimization criteria while meeting all constraints. But, since workflow scheduling is an NP-complete problem, it would take an enormous amount of time to find such solution for the complex workflow with more than one optimization criteria.

4 Proposed Solution

In our previous work [15] we have proposed genetic algorithm for joint evolution of two populations – resources configurations and mappings of tasks on resources – in a dynamically changing computational environment. This approach allows to greatly expand the search space for the optimal solution generation in order to efficiently adapt the environment for the changing requirements of the workload. General principle of the coevolutionary genetic algorithm (CGA) is demonstrated in Fig. 1.

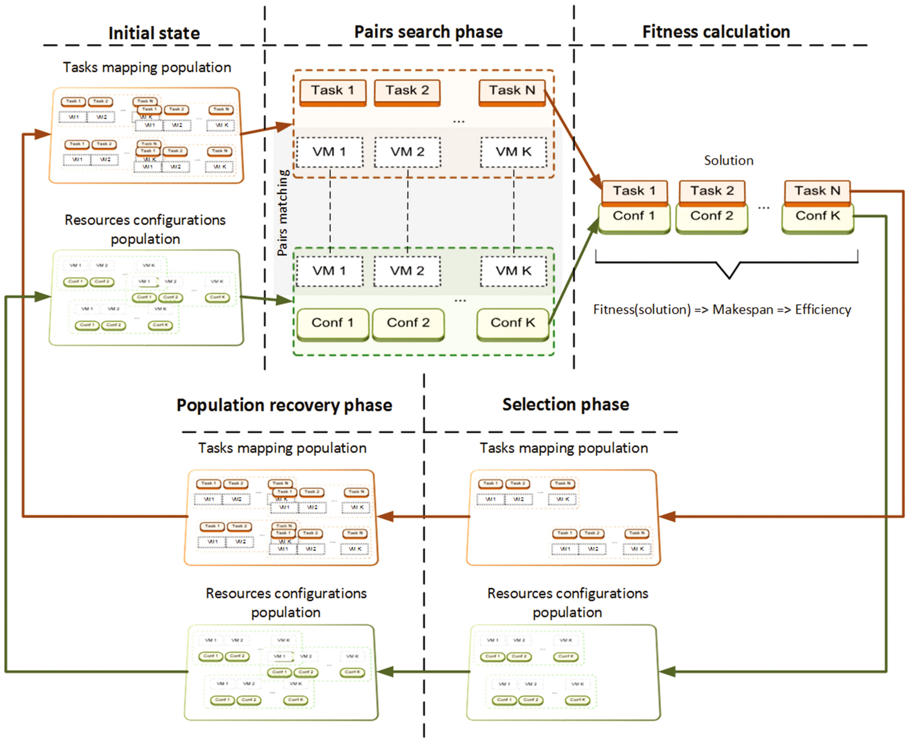

Fig. 1. General principle of CGA

CGA allows to produce an optimal configuration of virtual machines for the target workflow. But sometimes topology of the workflow requires dynamic reconfiguration of the resources for the maximum efficiency. In Fig. 2a sample workflow, which has six levels of tasks, is presented. In Fig. 2b we can see two schedules created for that workflow on a single resource with four CPU cores. For the sake of representability, we assume that the schedule shown in the top of Fig. 2b is the most optimal schedule obtained using CGA. It can be seen that in that schedule there are situations when virtual machines are staying idle, because parent tasks of the task mapped on that machines did not finish yet. To eliminate such situations, we propose to dynamically change configuration of the resources during the workflow execution. This can help to utilize computational resources to maximum effect. And while modern virtualization software allows to dynamically change properties of virtual machines on runtime, this approach brings little to no reconfiguration overheads. Example of the schedule generated with the use of dynamic reconfiguration is presented in the bottom of Fig. 2b.

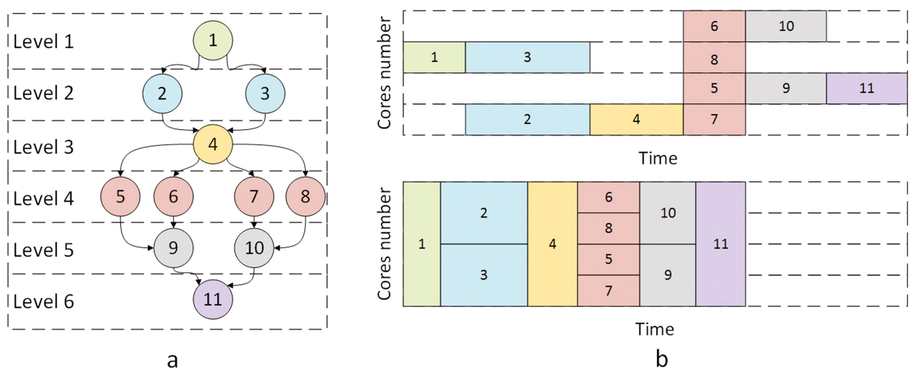

Fig. 2. Sample workflow (a) and its schedules (b)

To achieve the described effect, we have developed a dynamic resources coevolutionary genetic algorithm (DRCGA). Two main differences between DRCGA and CGA are representation of resources configuration chromosome and fitness function.

Representation of the resources configuration chromosome is presented in Fig. 3. The chromosome consists of independent reconfiguration schedules for all resources (R1 and R2 in Fig. 3). Every such schedule includes a set of time points (t0 – t4). Time point stores a required configuration of virtual machines. All reconfiguration schedules have initial time point t0, which cannot be changed or moved. In Fig. 3 reconfiguration is applied only to CPU cores, but the same method can be applied to other parameters, e.g. RAM volume.

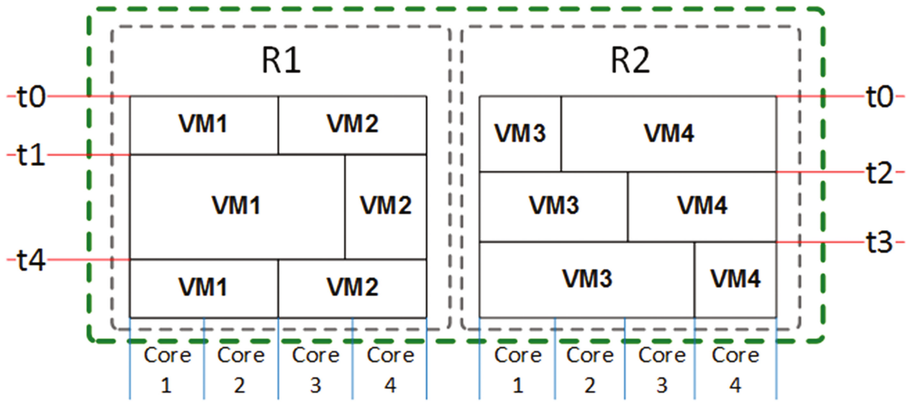

Fig. 3. Resources configuration chromosome representation

Before chromosomes generation maximal execution time is calculated as a sum of execution times for all tasks of the workflow on the least powerful node. Before the start of scheduling all chromosomes are initialized with randomly generated reconfiguration schedules – random set of time points with a random resources' configurations.

During the crossover operation child chromosome for every resource receives partitioning randomly selected from one parent or another. In Fig. 4 schema of crossover operation is presented. In the presented example child chromosome receives partitioning for R1 from the first parent and partitioning for R2 from the second parent.

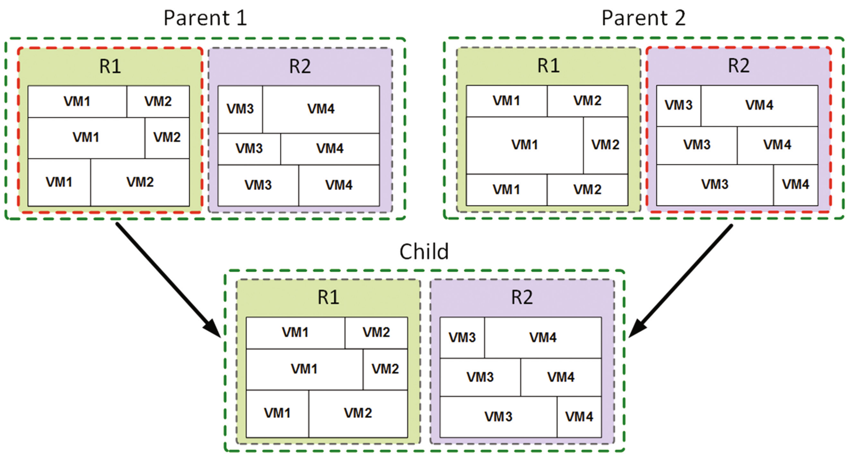

Fig. 4. Resources configuration crossover example

Mutation includes several scenarios of modifying resources configuration chromosome:

1. Transfer of capacity from one virtual machine to another. During this operation algorithm selects physical resource, two virtual machines located on this resource and the amount of capacity to transfer.
2. Addition of time point for the resource. Algorithm selects physical resource and adds new time point with a random configuration.
3. Removal of time point. Algorithm selects physical resource and removes some time point from it. This operation is available only if at least one resource of the chromosome has more than one time points.
4. Shift of time point. Algorithm selects a random time point for some resource and shifts upward or downward with regard to initial time point and maximal execution time.

Fitness function was also modified for proper consideration of dynamic characteristics of virtual machines. Fitness calculation models execution process. To do this algorithm takes tasks from the mapping chromosome one by one and estimates their execution time using performance models and virtual machine's parameters, which it has during the execution of the task. Algorithm also takes into account overheads of virtual machines reconfiguration.

5 Experimental Study

Experiments were conducted using our own workflow execution simulator. Computational environment is presented by a set of resources, each of which has a certain number of CPU cores and core capacity. Resources are partitioned into virtual machines. There are two types of network connections in the system – global (between resources) and local (between virtual machines of the same resource). Network bandwidth for local connections is much higher than for global ones – 100 Mbit/sec and 10 Mbit/sec respectively. Dynamic nature of the system was represented by probability of successful execution of the task on the virtual machine. This probability was set to 95% throughout the experiment. In case of the task failure virtual machine becomes unavailable for 60 s for the recovery.

In our work we used synthetic workflows, which replicate real scientific applications: Montage (astronomy), CyberShake (earthquake science) and Inspiral (gravitational physics) [16]. Workflows generator allows to create synthetic workflows of different sizes, which follow a general structure of the application. Synthetic workflows provide information about structure of the application, dependencies between tasks, data used for tasks execution and average execution time on benchmarking workstation. For the experimental evaluation of DRCGA following synthetic workflows were used – Montage 25, Montage 50, CyberShake 30, CyberShake 50, Inspiral 30 and Inspiral 50 (number in the name represents a number of tasks in the application).

Computational environment consists of three resources; every resource has four CPU cores. Initial distribution of cores between virtual machines in presented in Table 1.

Table 1. CPU cores distribution between virtual machines

	Resource 1	Resource 2	Resource 3
Virtual machine 1	1 core	2 cores	4 cores
Virtual machine 2	1 core	–	–
Virtual machine 3	1 core	2 cores	–
Virtual machine 4	1 core	–	–

Experimental evaluation was performed for four algorithms:

1. Heterogeneous Earliest Finish Time heuristic algorithm (HEFT) was used as a baseline for measuring quality of solutions.
2. Standard GA with one population and no resources reconfiguration.
3. CGA with two populations and static resources reconfiguration.
4. DRCGA with two populations and dynamic resources reconfiguration.

Parameters of metaheuristic algorithms are following:

- Crossover probability: 0.4
- Mutation probability: 0.4
- Population size: 100
- Iterations number: 2000

To provide statistical significance to the obtained results, all experiments were repeated 50 times. Results of experiments – average performance increase for 50 times – are presented in Table 2. As we can see, DRCGA outperforms other algorithms in all cases with the maximum performance increase 51.5%.

Table 2. Performance increase compared to HEFT (in %)

Workflow	Algorithm		
	GA	CGA	DRCGA
Montage 25	36.4	51.0	**51.4**
Montage 50	27.6	28.6	**33.3**
Montage 75	18.7	22.4	**30.5**
Montage 100	15.4	15.9	**24.9**
CyberShake 30	3.1	51.5	**51.5**
CyberShake 50	10.9	13.6	**32.8**
Inspiral 30	0.2	4.9	**15.6**
Inspiral 50	5.3	9.5	**17.0**

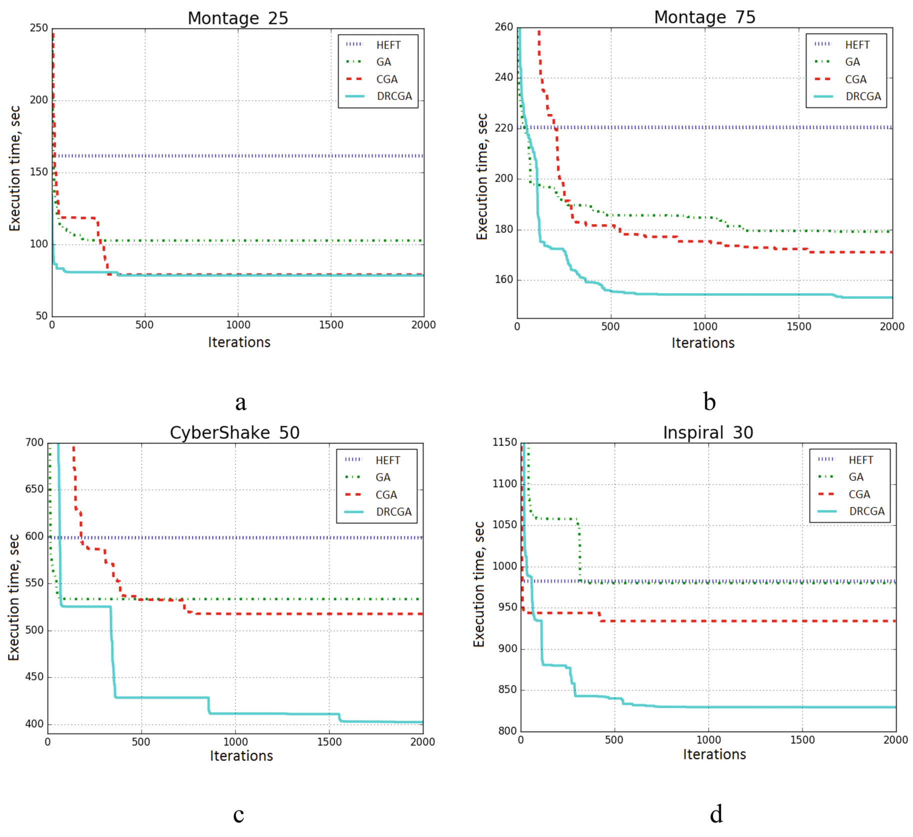

Fig. 5. Workflow execution time estimations change during algorithms processing

Advantage of wide search space of coevolutionary algorithms is demonstrated in Fig. 5. For small workflows (Figs. 5a and d) all algorithms converge to the optimum relatively fast (about 500 iterations), and coevolutionary algorithms require slightly more time to find the best solution. For larger workflows, on the other hand, it is likely for DRCGA to find more optimal solution even after many iterations – there are makespan improvements both for Montage 25 and CyberShake 50 after 1500 iterations.

6 Conclusion

In this paper we proposed a dynamic resources coevolution genetic algorithm (DRCGA), which extends previously developed approach of joint evolution of both computational environment configuration and mapping of tasks onto execution nodes. DRCGA utilizes virtualization mechanics to partition physical resources into virtual machines. Algorithm allows to reconfigure virtual machines on runtime to minimize idle time of the nodes, thus extending the search space and making possible to find more optimal solutions. DRCGA is able to handle changes in the computational environment, such as nodes failures. Algorithm was experimentally evaluated against HEFT, standard GA and DCGA in the dynamically changing simulation environment, where failures and other changes might occur. Results demonstrate that DRCGA generates better solutions for various types of scientific workflows and preserves capability to improve the solution for larger number of iterations than other metaheuristic algorithms.

Acknowledgements. This research financially supported by Ministry of Education and Science of the Russian Federation, Agreement #14.587.21.0024 (18.11.2015). Unique Identification RFMEFI58715X0024.

References

1. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Futur. Gener. Comput. Syst. **29**(3), 682–692 (2013)
2. Yu, J., Buyya, R.: A taxonomy of workflow management systems for grid computing. J. Grid Comput. **3**(3–4), 171–200 (2005)
3. Pugliese, R., Tiezzi, F.: A calculus for orchestration of web services. J. Appl. Logic **10**(1), 2–31 (2012)
4. Abrishami, S., Naghibzadeh, M., Epema, D.H.J.: Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Futur. Gener. Comput. Syst. **29**(1), 158–169 (2013)
5. Visheratin, A.A., Melnik, M., Nasonov, D.: Workflow scheduling algorithms for hard-deadline constrained cloud environments. Procedia Comput. Sci. **80**, 2098–2106 (2016)
6. Zhang, F., Cao, J., Li, K., Khan, S.U., Hwang, K.: Multi-objective scheduling of many tasks in cloud platforms. Futur. Gener. Comput. Syst. **37**, 309–320 (2014)
7. Arabnejad, H.: List Based Task Scheduling Algorithms on Heterogeneous Systems-An overview. Paginas.Fe.Up.Pt, p. 10 (2012)

8. Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., Kennedy, K.: Task scheduling strategies for workflow-based applications in grids. In: 2005 IEEE International Symposium on Cluster Computing and the Grid, CCGrid 2005, vol. 2, pp. 759–767 (2005)
9. Jakob, W., Strack, S., Quinte, A., Bengel, G., Stucky, K.-U., Süß, W.: Fast rescheduling of multiple workflows to constrained heterogeneous resources using multi-criteria memetic computing. Algorithms **6**(2), 245–277 (2013)
10. Nasonov, D., Melnik, M., Shindyapina, N., Butakov, N.: Metaheuristic coevolution workflow scheduling in cloud environment. In: IJCCI, vol. 1, pp. 252–260 (2015)
11. Liu, L., Zhang, M., Buyya, R., Fan, Q.: Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing. Concurr. Comput., 1–12 (2016)
12. Zhou, N., Qi, D., Wang, X., Zheng, Z., Lin, W.: A list scheduling algorithm for heterogeneous systems based on a critical node cost table and pessimistic cost table. Concurr. Comput. Pract. Exp. **29**(5) (2017). e3944
13. Abdulhamid, S.M., Abd Latiff, M.S., Abdul-Salaam, G., Hussain Madni, S.H.: Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. PLoS ONE **11**(7), e0158102 (2016)
14. Chen, H., Zhu, X., Qiu, D., Guo, H., Yang, L.T., Lu, P.: EONS: minimizing energy consumption for executing real-time workflows in virtualized cloud data centers. In: Proceedings International Conference Parallel Process Work, pp. 385–392 (2016)
15. Nasonov, D., Melnik, M., Radice, A.: Coevolutionary workflow scheduling in a dynamic cloud environment. In: International Conference on EUropean Transnational Education, pp. 189–200 (2016)
16. Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.H., Vahi, K.: Characterization of scientific workflows. In: 2008 3rd Workshop on Workflows in Support of Large-Scale Science, WORKS 2008 (2008)