Heuristic of Anticipation for Fair Scheduling and Resource Allocation in Grid VOs

Victor Toporkov, Anna Toporkova and Dmitry Yemelyanov

Abstract In this work, a job-flow scheduling approach for Grid virtual organizations (VOs) is proposed and studied. Users and resource providers preferences, VOs internal policies, resources geographical distribution along with local private utilization impose specific requirements for efficient scheduling according to different, usually contradictive, criteria. With increasing resources utilization level the available resources set and corresponding decision space are reduced. In order to improve overall scheduling efficiency, we propose an anticipation scheduling heuristic. It includes a target (anticipated) pattern solution definition and a special replication procedure for efficient and feasible resources allocation. A proposed anticipation algorithm is compared against conservative backfilling variations using such criteria as average jobs response time (start and finish times) as well as users and VO economic criteria (execution time and cost).

Keywords Scheduling · Grid · Utilization · Heuristic · Job batch · Virtual organization · Cycle scheduling scheme · Anticipation · Replication · Backfilling

1 Introduction and Related Works

In distributed environments with non-dedicated resources such as utility Grids the computational nodes are usually partly utilized by local high-priority jobs coming from resource owners. Thus, the resources available for use are represented with a set of slots—time intervals during which the individual computational nodes are

V. Toporkov (✉) · D. Yemelyanov
National Research University "MPEI", ul. Krasnokazarmennaya, 14,
Moscow 111250, Russia
e-mail: ToporkovVV@mpei.ru

D. Yemelyanov
e-mail: YemelyanovDM@mpei.ru

A. Toporkova National Research University Higher School of Economics,
ul. Myasnitskaya, 20, Moscow 101000, Russia
e-mail: AToporkova@hse.ru

M. Ivanović et al. (eds.), *Intelligent Distributed Computing XI*,
Studies in Computational Intelligence 737, https://doi.org/10.1007/978-3-319-66379-1_3

capable to execute parts of independent users' parallel jobs. These slots generally have different start and finish times and a performance difference. The presence of a set of slots impedes the problem of coordinated selection of the resources necessary to execute the job-flow from computational environment users. Resource fragmentation also results in a decrease of the total computing environment utilization level [1, 2].

Two established trends may be outlined among diverse approaches to distributed computing. The first one is based on the available resources utilization and application level scheduling [3]. As a rule, this approach does not imply any global resource sharing or allocation policy. Another trend is related to the formation of user's virtual organizations (VO) and job-flow scheduling [4, 5]. In this case a metascheduler is an intermediate chain between the users and local resource management and job batch processing systems.

Uniform rules of resource sharing and consumption, in particular based on economic models, make it possible to improve the job-flow level scheduling and resource distribution efficiency. VO policy may offer optimized scheduling to satisfy both users' and VO common preferences. The VO scheduling problems may be formulated as follows: to optimize users' criteria or utility function for selected jobs [6, 7], to keep resource overall load balance [8, 9], to have job run in strict order or maintain job priorities [10], to optimize overall scheduling performance by some custom criteria [11, 12], etc.

VO formation and performance largely depends on mutually beneficial collaboration between all the related stakeholders. Thus, VO policies in general should respect all members and the most important aspect of rules suggested by VO is their fairness.

A number of works understand fairness as it is defined in the theories of cooperative games and mechanism design, such as fair job-flow distribution [8], fair quotas [13, 14] or fair user jobs prioritization [10]. The cyclic scheduling scheme (CSS) [15] implements a fair scheduling optimization mechanism which ensures stakeholders interests to some predefined extent. Thus, we elaborate a problem of parallel jobs scheduling in heterogeneous computing environment with non-dedicated resources considering users individual preferences and goals.

The downside of a majority centralized metascheduling approaches is that they lose their efficiency and optimization features in distributed environments with a limited resources supply. For example, in [2] a traditional backfilling algorithm provides better scheduling outcome when compared to different optimization approaches in resource domain with a minimal performance configuration. The general root cause is that in fact the same scarce set of resources (being efficient or not) have to be used for a job-flow execution or otherwise some jobs might hang in the queue. And under such conditions user jobs priority and ordering greatly influence the scheduling results.

A main contribution of this paper is a heuristic anticipation job-flow scheduling approach which retains optimization features and efficiency even in distributed computing environments with limited resources. The rest of the paper is organized as follows. Section 2 presents a general CSS fair scheduling concept. The proposed anticipation scheduling technique is presented in Sect. 3. Section 4 contains simulation experiment setup and results of comparison with conservative backfilling variations. Finally, Sect. 5 summarizes the paper.

2 Cyclic Alternative-Based Fair Scheduling Model and Limited Resources

Scheduling of a job-flow using CSS is performed in time cycles known as scheduling intervals, by job batches [15]. The actual scheduling procedure during each cycle consists of two main steps. The first step involves a search for alternative execution scenarios for each job or simply alternatives [16]. During the second step the dynamic programming methods [15] are used to choose an optimal alternatives' combination (one alternative is selected for each job) with respect to the given VO and user criteria. This combination represents the final schedule based on current data on resources load and possible alternative executions.

An example for a user scheduling criterion may be a minimization of overall job running time, a minimization of overall running cost, etc. This criterion describes user's preferences for that specific job execution.

Alongside with time (T) and cost (C) properties each job execution alternative has a user utility (U) value: user evaluation against the scheduling criterion. We consider a relative approach to represent a user utility $U \in [0\%, 100\%]$. Each alternative gets its utility in relation to the "best" and the "worst" optimization criterion values user could expect according to the job's priority. And the more some alternative corresponds to user's preferences the smaller is the value of $U \to 0\%$.

For a fair scheduling model the second step VO optimization problem could be in form of: $C \to \max$, $\lim U$, i.e. maximize total job-flow execution cost, while respecting user's preferences to some extent.

First step of CSS requires allocation of a multiple alternatives *nonintersecting* in terms of slots for each job. Otherwise irresolvable collisions for resources may occur if different jobs will share the same time-slots. Sequential alternatives search and resources reservation procedures help to prevent such scenario. However in an extreme case when resources are limited or overutilized only at most one alternative execution could be reserved for each job. In this case alternatives-based scheduling result will be no different from FIFO resources allocation procedure without any optimizations [2].

3 Heuristic Anticipation Scheduling

3.1 General Scheme

In order to improve scheduling efficiency for job batch the following heuristic is proposed. It consists of three main steps.

1. First, a set of all possible execution alternatives is found for each job not considering time slots intersections and without any resources reservation.
2. Second, CSS scheduling procedure is performed to select alternatives combination (one alternative for each job of the batch) optimal according to VO fair-share policy. The resulting alternatives combination most likely corresponds to an infeasible scheduling solution as possible time slots intersection will cause collisions on resources allocation stage.
3. Third, a feasible resources allocation is performed by replicating alternatives selected in step **2**.

After these three steps are performed the resulting solution is both feasible and efficient as it reflects scheduling pattern obtained from a near-optimal reference solution from step **2**. The following subsections will discuss these scheduling steps in more details.

3.2 Finding a Near Optimal Infeasible Scheduling Solution

CSS scheduling results are strongly depend on diversity of alternatives sets obtained for batch jobs. As we need to find alternatives for an apriori infeasible reference solution a reasonable diverse set of possible execution alternatives will do.

We used a modification of Algorithm searching for Extreme Performance windows (AEP) [16] to allocate a diverse set of execution alternatives for each job. Originally AEP scans through a whole list of available time slots and retrieves one alternative execution optimal according to the user custom criterion. During this scan, AEP estimates every possible and sufficient slots combination against user criterion and selects the one with the best criterion value. In order to retrieve all possible execution alternatives we save all distinct intermediate AEP search results to a dedicated list of possible alternatives.

After sets of possible execution alternatives are independently allocated for each job a CSS scheduling optimization procedure selects an optimal alternatives combination according to VO and users criteria [15]. More details on alternatives combination selection procedure were provided in Sect. 2.

3.3 Replication Scheduling and Resources Allocation

The resulting near-optimal scheduling solution in most cases is infeasible as selected alternatives may share the same time slots and thus cause resource collisions. However we propose to use it as a reference solution and replicate into a feasible resources allocation.

For the replication purpose a new *Execution Similarity* criterion is introduced. It helps AEP [16] to find a window with minimum *distance* to a reference pattern alternative. Generally we define a *distance* between two different alternatives (windows) as a relative difference or *error* between their significant criteria values. For example if reference alternative has C_{ref} total cost, and some candidate alternative cost is C_{can}, then the relative cost error E_C is calculated as $E_C = \frac{|C_{\text{ref}} - C_{\text{can}}|}{C_{\text{ref}}}$. If one need to consider several criteria the *distance D* between two alternatives may be calculated as a geometric distance in a parameters space: $D_g = \sqrt{E_C^2 + E_T^2 + \cdots + E_U^2}$.

AEP with *Execution Similarity* scans through a whole list of available time slots, for every possible slots combination calculates its distance from a reference alternative and selects the one with the minimum distance to a reference.

For a feasible job batch resources allocation AEP consequentially allocates for each job a single execution window with a minimum *distance* to a reference alternative. Time slots allocated for the i-th job are reserved and excluded from the slot list when AEP search algorithm is performed for the following jobs $i + 1, i + 2, \ldots$. Thus, this procedure prevents any conflicts for resources and provides scheduling solution which in some sense reflects near-optimal reference solution.

AEP and its modifications have a quadratic computational complexity with respect to the number of available computing nodes [16]. It is performed twice for each job (during alternatives search and replication steps), so the overall complexity linearly depends on the job-flow capacity. At the same time dynamic scheduling scheme [15] from the step **2** is pseudo-polynomial and additionally depends on a total budget allocated for the job-flow execution.

3.4 Replication Reference Setup

Anticipated near-optimal scheduling solution provides a heuristic insight on how each job should be executed with a reference to other users criteria, VO optimization policy and a current computing domain composition and utilization level. Basically this solution suggests what kind of resources should be allocated for each job in terms of performance and cost. Thus, available resources can be consistently distributed between the user jobs according to their performance or cost optimization targets.

At the same time the anticipated solution can't provide any meaningful reference on jobs' start and finish times. As anticipation procedure independently allocates a set of possible execution alternatives for each job, it does not consider resources

reservation and utilization by other jobs. Thus, resulting anticipated jobs' start and finish times are randomly distributed on a whole scheduling interval with a bias towards the interval's start. In this way anticipation scheduling scheme can't provide neither adequate estimation on jobs' starting times, nor the common jobs' execution order.

In order to improve the anticipated reference solution we use backfilling algorithm to provide practical values for jobs start and finish times. Backfilling is able to minimize the whole job-flow execution makespan as well as to generally follow the initial jobs relative queue order [1]. These features make backfilling scheduling solution a good reference target for the anticipation scheduling scheme. Thus, for the replication step we set infeasible CSS solution as a reference for jobs execution runtime and cost, and backfilling solution—for jobs start and finish times.

Additionally we introduce a finish time approximation coefficient K_t to relate the anticipated finish times to backfilling reference solution. For example when $K_t = 1$ we use exact jobs finish times provided by backfilling as a reference for a replication step. $K_t = 0.5$ means that we strive to execute the job-flow twice as faster compared to backfilling. So just by changing K_t we are able stretch resulting anticipation solution on a desired time interval and at the same time preserve a general jobs execution order provided by backfilling.

4 Simulation Study

4.1 Simulation Environment Setup

An experiment was prepared as follows using a custom distributed environment simulator [2, 15–17]. For our purpose, it implements a heterogeneous resource domain model: nodes have different usage costs and performance levels. A space-shared resources allocation policy simulates a local queuing system (like in GridSim or CloudSim [18]) and, thus, each node can process only one task at any given simulation time. The execution cost of each task depends on its running time which is proportional to the dedicated nodes performance level. The execution of a single job requires parallel execution of all its tasks.

Virtual organization and computing environment properties are the following. The resource pool includes 25 heterogeneous computational nodes. A base cost of a node is an exponential function of its performance value, so any two nodes of the same resource type and performance have the same base cost. Effective node cost during the scheduling interval is then calculated by adding a variable distributed normally as ± 0.6 of a base cost, simulating discounts or extra charges up to 60%. The initial 5–10% resource load with owner jobs is distributed hyper-geometrically over the whole scheduling interval.

The job batch properties are specified as follows. Jobs number in a batch is 75. Nodes quantity needed for a job is an integer number distributed evenly on $[2, 5]$.

Node reservation time is an integer number distributed evenly on [100, 600]. Job budget varies in the way that some of jobs can pay as much as 160% of base cost whereas some may require a discount. Every request contains a specification of a custom criterion which is one of the following: *job execution runtime* or *overall execution cost*.

During each experiment a VO resource domain and a job batch were generated and the following scheduling algorithms were simulated and studied.

First we ran a conservative backfilling algorithm BF_s to obtain an exemplary job-flow scheduling solution. Conservative backfilling consequently starts each job as soon as possible on condition it does not delay execution of higher priority jobs. Next, we ran a conservative backfilling modification BF_f, which instead of minimizing jobs' start times, performs jobs' finish time minimization with the same restriction to delay high priority jobs. For this purpose we used AEP algorithm with a finish time minimization criterion to find and allocate suitable resources for each job.

Finally we performed anticipation scheduling procedure *ANT* with a $C \rightarrow \max$, $\lim U_a = 10\%$ policy and different approximation coefficient values $K_t \in \{0, 0.1, 0.5, 1, 1.1, 1.5, \}$ (see Sect. 3.4).

4.2 Simulation Results

More then 2000 scheduling cycles were simulated to obtain average job-flow scheduling results for BF_s, BF_f and *ANT*. Figures 1 and 2 show average job-flow starting and finishing times as a function of K_t parameter.

First it should be noted that BF_f algorithm at average provided 2% earlier jobs start times and 7% earlier finish times compared to a simple BF_s implementation. Thus, considered backfilling modification BF_f provides an even higher scheduling standard for an anticipation scheme. At the same time *ANT* provided earlier jobs

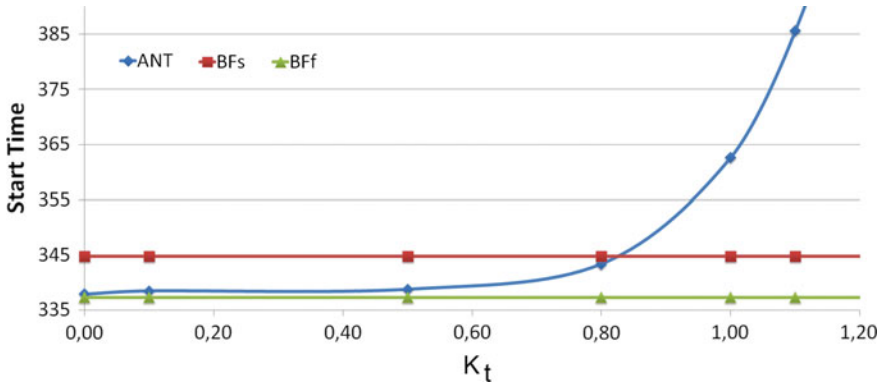

Fig. 1 Simulation results: average job execution start time

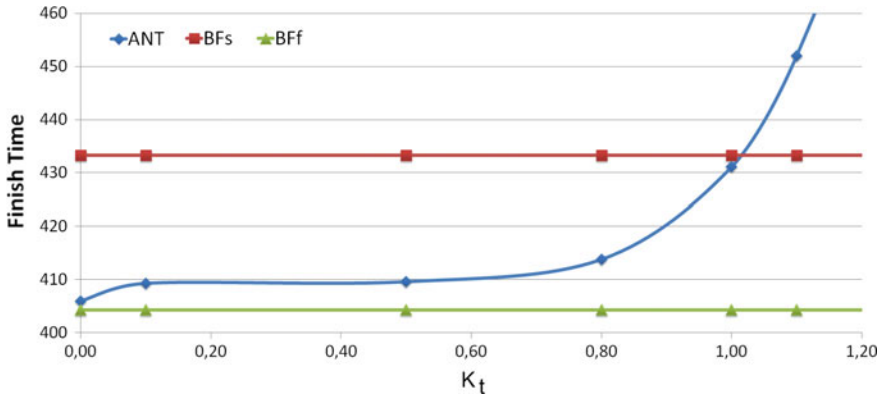

Fig. 2 Simulation results: average job execution finish time

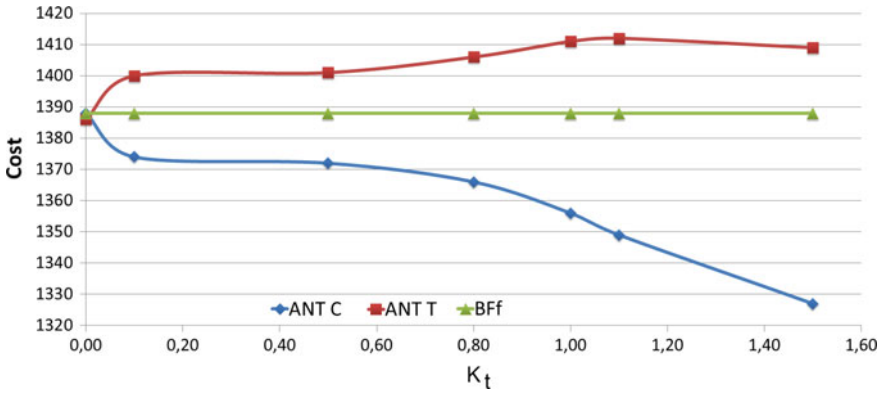

Fig. 3 Simulation results: average job execution cost

finishing times compared to BF_s for all $0 < K_t \leq 1$. When $K_t > 1$ by K_t definition *ANT* jobs finish times are as expected longer then in a reference backfilling solution.

It can be observed on Figs. 1 and 2 that by decreasing K_t in *ANT* job-flow average start and finish times are decreasing and tends to BF_f result. In an extreme case when $K_t = 0$ and no job-flow optimization is performed, BF_f advantage is less then 1%. With $0 < K_t \leq 1$ values *ANT* with a 2–7% longer job-flow finishing time is still able to perform job-flow scheduling optimization (Figs. 3 and 4).

Figure 3 shows average jobs execution cost for jobs with a cost minimization (ANT_C) and runtime minimization (ANT_T) criteria obtained by *ANT* and BF_f. As expected with $K_t = 0$ ANT_C, ANT_T and BF_f have the same jobs execution cost as no job-flow optimization is performed by *ANT*. However when $0 < K_t \leq 1$ *ANT* allocates resources according to scheduling policies and hence ANT_C jobs has 1–2% less execution cost compared to backfilling and 2–4% less compared to ANT_T. A similar picture is presented on Fig. 4 for an average jobs' execution runtime. With a rela-

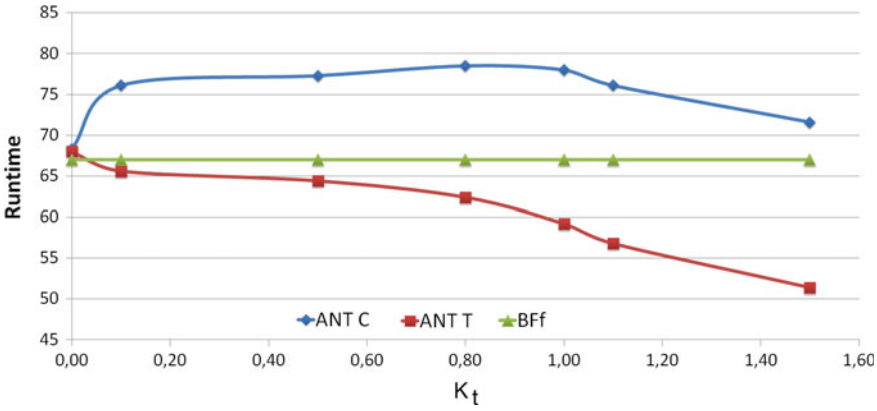

Fig. 4 Simulation results: average job execution runtime

tively small values $K_t < 0.8\, ANT_T$ provides up to 6% shorter jobs runtime compared to backfilling and 20% shorter compared to ANT_C jobs. With increasing K_t ANT_T advantage over backfilling increases and reaches 22% when $K_t = 1.5$.

Summarizing the results, anticipation scheduling algorithm is able to perform an efficient and fair resources allocation and provide competitive job-flow execution completion time. This achieved by a replication procedure which uses backfilling and CSS scheduling results combination as a reference target solution.

5 Conclusions and Future Work

In this paper we study the problem of a fair job batch scheduling with a relatively limited resources supply. We introduce a heuristic scheduling scheme which uses combination of a fair share scheduling policy with a common backfilling algorithm as a reference to allocate a feasible accessible solution.

Computer simulation was performed to study anticipation scheduling scheme and to evaluate its efficiency. The obtained results show that the new heuristic approach provides flexible solutions for different fair scheduling scenarios while job-flow execution time is only 2–7% longer compared to backfilling.

Future work will be focused on replication algorithm study and its possible application to fulfil complex user preferences expressed in a resource request.

Acknowledgements This work was partially supported by the Council on Grants of the President of the Russian Federation for State Support of Young Scientists and Leading Scientific Schools (grants YPhD-2297.2017.9 and SS-6577.2016.9), RFBR (grants 15-07-02259 and 15-07-03401) and by the Ministry on Education and Science of the Russian Federation (project no. 2.9606.2017/8.9).

References

1. Dimitriadou, S.K., Karatza, H.D.: Job scheduling in a distributed system using backfilling with inaccurate runtime computations. In: Proceedings of 2010 International Conference on Complex, Intelligent and Software Intensive Systems, pp. 329-336 (2010). doi:10.1109/CISIS.2010.65
2. Toporkov, V., Toporkova, A., Tselishchev, A., Yemelyanov, D., Potekhin, P.: Heuristic strategies for preference-based scheduling in virtual organizations of utility grids. J. Ambient Intell. Hum. Comput. **6**(6), 733–740 (2015). doi:10.1007/s12652-015-0274-y
3. Buyya, R., Abramson, D., Giddy, J.: Economic models for resource management and scheduling in grid computing. J. Concurrency Comput. **14**(5), 1507–1542 (2002). doi:10.1002/cpe.690
4. Kurowski, K., Nabrzyski, J., Oleksiak, A. and Weglarz, J.: Multicriteria aspects of grid resource management. In: Nabrzyski, J., Schopf, J.M. and Weglarz, J. (eds.) Grid Resource Management. State of the Art and Future Trends, pp. 271–293 (2003). doi:10.1007/978-1-4615-0509-9_18
5. Rodero, I., Villegas, D., Bobro, N., Liu, Y., Fong, L., Sadjadi, S.M.: Enabling interoperability among grid meta-schedulers. J. Grid Comput. **11**(2), 311–336 (2013). doi:10.1007/s10723-013-9252-9
6. Ernemann, C., Hamscher, V., Yahyapour, R.: Economic scheduling in grid computing. In: Revised Papers from the 8th International Workshop on Job Scheduling Strategies for Parallel Processing, vol. 2537, pp. 128–152. Springer, Berlin, Heidelberg (2002). doi:10.1007/3-540-36180-4_8
7. Rzadca, K., Trystram, D., Wierzbicki, A.: Fair game-theoretic resource management in dedicated Grids. In: IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2007), pp. 343–350 (2007). doi:10.1109/ccgrid.2007.52
8. Penmatsa, S., Chronopoulos, A.T.: Cost minimization in utility computing systems. Concurrency Comput.: Pract. Experience **16**(1), 287–307 (2014). doi:10.1002/cpe.2984
9. Vasile, M., Pop, F., Tutueanu, R., Cristea, V., Kolodziej, J.: Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. J. Future Gener. Comput. Syst. **51**, 61–71 (2015). doi:10.1016/j.future.2014.11.019
10. Mutz, A., Wolski, R. and Brevik, J.: Eliciting honest value information in a batch-queue environment. In: 8th IEEE/ACM International Conference on Grid Computing, pp. 291–297, IEEE Computer Society (2007). doi:10.1109/grid.2007.4354145
11. Blanco, H., Guirado, F., Lrida, J.L., Albornoz, V.M.: MIP model scheduling for multi-clusters. Proc. Euro-Par **2012**, 196–206 (2012). doi:10.1007/978-3-642-36949-0_22
12. Takefusa, A., Nakada, H., Kudoh, T., Tanaka, Y.: An advance reservation-based co-allocation algorithm for distributed computers and network bandwidth on QoS-guaranteed grids. In: 15th International Workshop JSSPP 2010, vol. 6253, pp. 16–34 (2010). doi:10.1007/978-3-642-16505-4_2
13. Carroll, T., Grosu, D.: Divisible load scheduling: an approach using coalitional games. In: Proceedings of the Sixth International Symposium on Parallel and Distributed Computing (ISPDC 07), pp. 36–36 (2007). doi:10.1109/ispdc.2007.16
14. Kim, K., Buyya, R.: Fair resource sharing in hierarchical virtual organizations for global grids. In: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, pp. 50–57 (2007). doi:10.1109/grid.2007.4354115
15. Toporkov, V., Yemelyanov, D., Bobchenkov, A., Tselishchev, A.: Scheduling in Grid Based on VO Stakeholders Preferences and Criteria. Advances in Intelligent Systems and Computing, vol. 470, pp. 505–515. Springer International Publishing Switzerland (2016). doi:10.1007/978-3-319-39639-2_44
16. Toporkov, V., Toporkova, A., Tselishchev, A., Yemelyanov, D.: Slot selection algorithms in distributed computing. J. Supercomput. **69**(1), 53–60 (2014). doi:10.1007/s11227-014-1210-1

17. Toporkov, V., Tselishchev, A., Yemelyanov, D., Bobchenkov, A.: Composite scheduling strategies in distributed computing with non-dedicated resources. Proc. Comput. Sci. **9**, 176–185 (2012). doi:10.1016/j.procs.2012.04.019
18. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. J. Softw.: Pract. Experience **41**(1), 23–50 (2011). doi:10.1002/spe.995