Survey article

Ilya Chernov, Natalia Nikitina*, and Evgeny Ivashko

Task Scheduling in Desktop Grids: Open Problems

https://doi.org/10.1515/eng-2017-0038
Received September 7, 2017; accepted November 8, 2017

Abstract: We survey the areas of Desktop Grid task scheduling that seem to be insufficiently studied so far and are promising for efficiency, reliability, and quality of Desktop Grid computing. These topics include optimal task grouping, "needle in a haystack" paradigm, game-theoretical scheduling, domain-imposed approaches, special optimization of the final stage of the batch computation, and Enterprise Desktop Grids.

Keywords: desktop grid, task scheduling, high-performance computing, high-throughput computing, enterprise desktop grid

1 Introduction

Desktop Grid is a computational paradigm representing a distributed computing system which uses idle time of non-dedicated geographically distributed computing nodes connected over general-purpose network. Fast development of the Internet, quick growth of the number of personal computers, as well as huge increase in their performance, have made Desktop Grid a promising high-throughput computing tool for solving numerous scientific problems. Volunteer computing is one way to organize computation: the work is done by volunteers that join the project via the Internet. Another approach is Enterprise Desktop Grid which joins computers within an organization.

Ilya Chernov: Institute of Applied Mathematical Research of the Karelian Research Center RAS; Petrozavodsk State University, Petrozavodsk, Russia, E-mail: IAChernov@yandex.ru
***Corresponding Author: Natalia Nikitina:** Institute of Applied Mathematical Research of the Karelian Research Center RAS, Petrozavodsk, Russia, E-mail: nikitina@krc.karelia.ru
Evgeny Ivashko: Institute of Applied Mathematical Research of the Karelian Research Center RAS; Petrozavodsk State University, Petrozavodsk, Russia, E-mail: ivashko@krc.karelia.ru

The first large volunteer computing project SETI@home was launched in 1999, providing the basis for development of the BOINC (Berkeley Open Infrastructure for Network Computing) platform. Today, there are more than 60 active BOINC projects utilizing more than 15 million computers worldwide [2]. Desktop Grid differs from other high-performance systems, such as Computational Grids, computing clusters, and supercomputers; scheduling policies should be aware of huge hardware and software heterogeneity, lack of trust, uncertainty on availability and reliability of computing nodes, etc.

Note that, besides the conventional Desktop Grids, special types of them are considered in literature, including hybrid, hierarchical, peer-to-peer Desktop Grids, etc. Here, we address all variations as Desktop Grids. However, we focus on the classical BOINC architecture: client-server network with independent nodes, free to join or leave.

Hierarchical, peer-to-peer and decentralized Desktop Grids open new problems and challenges from the point of view of scheduling. The authors of [9], for example, propose a decentralized distributed scheduling method with each node serving as a computing node, scheduler, and router; in [20, 21, 27] (to mention a few) "Desktop Grids of Desktop Grids" are considered, where the nodes of a Desktop Grid are Desktop Grids with their own schedulers; in [43] scheduling in peer-to-peer Desktop Grids is studied. Also, new technologies and approaches are used to create new types of Desktop Grids. For example, the quickly developing blockchain technology is used to construct a market of Desktop Grid resources based on XtremWeb-HEP software platform (http://iex.ec/). These technologies and new approaches, together with scheduling policies and open problems that appear in this area, deserve a whole separate paper.

Scheduling policy used in Desktop Grid significantly affects its characteristics, such as performance (measured as the makespan, overall throughput, throughput of validated results, turnaround, or otherwise) and/or reliability (in the sense of producing correct results). So, much attention has been paid to development of effective (from different points of view) scheduling policies. However, there remain open problems that deserve more attention of researchers and are still promising.

Besides performance or reliability, one can consider the problem of reducing the cost of an answer. In particular, this can be restricted to energy efficiency (e.g., [34, 38, 40, 51]).

The aim of our work is to highlight several open problems in the domain of scheduling in Desktop Grid. Implementing the solutions of these problems promises significant performance improvement both in the common computational process and in specific domain-related problems. Our analysis is based on study of more than a hundred research papers on Desktop Grid scheduling and related problems, dated from 1999 to 2016. The comprehensive review of these papers is presented in [11]: we summarize methods of solving various challenges of task scheduling, but do not discuss open problems.

There is a number of articles devoted to overview and study of scheduling in Desktop Grids. The BOINC website has a list of open problems [3] (however, mostly technical ones). The paper [32] studies BOINC from the user and developer perspectives to develop high-level suggestions to future development of BOINC.

There is also a number of review papers devoted to Desktop Grid scheduling and related problems. The report [14] presents classifications of Desktop Grids from several points of view. Although relatively old, this work is still a valuable attempt to classify Desktop Grids and a collection of real systems.

The authors of [28] consider and compare the most popular contemporary middleware systems for Desktop Grid: BOINC, XtremWeb, OurGrid and HTCondor. They review scientific papers devoted to research of the factors that influence the performance of Desktop Grid (from the point of view of both server and the client).

The review [57] considers Computational Grids, though offers much knowledge about scheduling types and methods. Computational Grids, e-science Grids, Service Grids are all quite different compared to Desktop Grids.

The chapter [19] in a book is a review of scheduling algorithms for Desktop Grids and the challenges they face. Naive algorithms that ignore the work history, knowledge-based ones that use accumulated and apriori information, and adaptive methods are overviewed.

The paper [17] addresses availability of resources: the problem of great importance for volunteer computing. Measurement of resource availability, revealing availability patterns, and using them for better control of the computing process are the main considered questions. Several naive and knowledge-based algorithms are reviewed together with node grouping methods, security threats in volunteer computing, and safety measures.

In this paper, we focus on the open problems on task scheduling in BOINC. Using the BOINC terminology [4], by *task* we mean an instance of a computational workunit. The open problems we discuss in this paper include: task grouping and scheduling parcels of tasks; task scheduling when searching for rare answers; evaluating schedules from the point of view of the mean cost; non-common, domain-imposed scheduling objectives; game-theoretic methods of task scheduling; Enterprise Desktop Grids; "tail" phase scheduling; and generating policies by a complex "black box" algorithm, the inner logic of which is unavailable (including genetic search and neural networks).

2 Task grouping

By its essence, BOINC is designed to support applications that have low data transmission/compute ratio [1]. Thus, an important concern is providing computational tasks of reasonable runtime. On the one hand, too long tasks lead to waste of resources due to computational errors and deadline violations. It is also unfavourable from the point of view of volunteer computing [7]. On the other hand, too short tasks lead to high server load because of large number of requests from the clients.

Some scientific problems, like search for extrema in high-dimensional spaces, allow to vary task runtimes quite flexibly. But in some cases, selection of the proper task size is more complicated. One of such cases is virtual drug screening, where a single estimation of ligand-protein interaction energy typically takes from a couple of minutes to about an hour using molecular docking software such as AutoDock Vina [23, 42]. Task grouping technique could be used to solve this problem.

We will refer to a group of tasks wrapped together as a *parcel*. A parcel is created and processed as a single BOINC workunit. This distinguishes the parcel concept from a batch of tasks (which is a series of workunits) and from a group of tasks which is an arbitrarily selected portion of workunits.

Task parceling can serve at least three purposes. Firstly, parcels of small tasks take more time to solve, and communications between nodes and the server are fewer; so the server load is less. We witnessed a drastic slow-down of Desktop Grid computation when many quick tasks (less than 1 minute each) were scheduled; grouping tasks into parcels of 1000 tasks each improved the performance. In [37] the server load is minimized by choosing optimal parcel size by game-theoretic methods.

Next, parceling is able to improve scheduling efficiency by using variable size of parcels depending on a node's performance, reliability, trust level, network connection, etc. This question has been paid relatively low attention, up to our knowledge. In [35] different heuristic scheduling algorithms, including a few "batch" ones, are compared. Beside selecting the best ones by simulation, they also show that the batch mode allows more effective scheduling.

Finally, parcels can contain special tasks for revealing cheaters, checking purposes, etc. Adding some tasks of a parcel to another parcel for check purposes can be called "fractional replication": only part of the complex compound task is re-solved. This can be used to fight saboteurs, including intellectual and organized ones, to improve reliability, or for better efficiency. For example, [50] studies defense against intellectual cheaters (adversaries) in volunteer computing networks by putting check tasks into task parcels. In [58] also test tasks are added to parcels, not only to reveal the cheaters, but to hide valuable answers. The Earth added to a parcel of planet descriptions for search of life, drastically increases the amount of positive results so that stealing a positive result would be harder. In [16] the same aim is considered from another point of view: they re-solve randomly chosen tasks of a parcel to reveal lying nodes. Different retrieval methods for better performance are compared in [52]: this is not exactly task parceling, but a related approach.

3 Needle in a haystack

Many problems suitable for Desktop Grid consist of tasks that give interesting results only seldom. For example, fitting experimental data of hydride decomposition [24] by model curves by search in a space of parameters rejects most of the tested sets. Then errors of the first and second kinds, i.e., accepting a wrong answer or rejecting the correct one, have significantly different value. Indeed, a false interesting result would be quickly revealed; also, additional checks, even expensive, are too rare to significantly affect the metrics. However, rejecting the unique interesting result makes all the project completely useless. This can be taken into account for optimizing replication, scheduling, task validation, etc.

Besides, the existing paradigm "one task — one answer" obviously causes needless communications between the server and the clients. It seems effective to schedule huge parcels of tasks and consider the first good answer as the result of the whole parcel. In other words, a node continues to look for a good answer until either the parcel is complete or the deadline is missed. Up to our knowledge, nobody has studied this possibility which can be called "looking for the needle in a haystack". In volunteer computing, some reliability-improving measures must be taken. Heartbeat and replication techniques [49] can be used for this purpose. For Enterprise Desktop Grids, employment of the PUSH model of interaction with computing nodes may improve efficiency and significantly reduce the server load.

4 Cost of an answer

In [10, 12] we consider the special optimal replication problem for Desktop Grid: the minimized quantity is the average total cost of solving a task. Each task can produce an answer from some given set of values (a two-element set in case of recognition problems). Wrong answers can be obtained with some known probabilities. To reduce the risk of error, each task is solved independently on different computing nodes. An answer is accepted when it is produced the given number of times called the quorum (it can be answer-dependent). In case a wrong answer is accepted, some penalty is added to the computational cost. Penalties can also depend on the accepted answer and the correct answer and are usually large compared to the cost of a single computation. So, the cost of a task, consists of solving it several times (replication) and a possible penalty. The problem is to choose quorums for the possible answers to minimize the expected cost. Too high quorums almost eliminate the risk of paying a penalty but instead increase linearly the computational cost.

Obviously, the optimal quorums need not to be the same for different possible answers: some answers need to be checked more carefully than others, depending on their penalty and error probability. For example, if penalties are significantly different (so some answers are more valuable than other), less valuable answers need to be checked more in order to reduce the risk of missing a valuable answer. Another result is the stability of the optimal quorum with respect to the penalty values: critical penalties in case of reliable (i.e., with low error risk p) computing nodes grow as $p^{-\nu}$ with respect to the quorum ν. Therefore, exact values of penalties need not to be known precisely: rough estimations up to a few orders of magnitude are sufficient. Also the quorum for rare results often can be chosen higher than the optimal value at a very low cost.

Indeed, higher quorum further reduces the risk, but increases computational costs linearly; this addition occurs only seldom: when the true answer is a rare one. So the total increase of the average total cost is low.

5 Domain-imposed criteria for task scheduling

Desktop Grids allow solving problems from various scientific domains. Beside the common optimization criteria, such as the minimal makespan, the maximal throughput, etc., the scientific domain of the problem being solved on a Desktop Grid can impose additional requirements and optimization objectives to the process of problem solution, which partly might be addressed by proper task scheduling.

One of the examples of such objectives arises from the bioinformatics field. During virtual drug screening, the challenge is not only to process large libraries of molecules, but also to select a topologically diverse set of hit compounds. Moreover, such set is highly desirable to obtain at early stages of the research, because the virtual screening process might take months before the results proceed to a laboratory [45].

The work [39] proposes a task scheduling algorithm to address this challenge. The space of molecules is divided into blocks according to the chemical properties of molecules. The client nodes of a Desktop Grid act as independent entities and select a block where they prefer to perform molecular docking. The purpose of each node is to obtain as many hit molecules as possible, but topologically different from the ones found by other nodes. The prototype implementation and computational experiments show that the proposed algorithm allows to quickly obtain topologically diverse hits.

The example above shows that domain-imposed criteria for task scheduling can significantly increase effectiveness of a Desktop Grid.

6 Games of scheduling

Mathematical game theory is an effective approach to construct optimal rules of interaction between independent agents. In Desktop Grids, computing nodes can be considered as independent agents with different characteristics (i.e. computing power, network speed, availability and reliability metrics, and so on).

Game theory considers optimization problems where utility functions of two or more agents called players depend on decisions made by all players [54]. One of the aims of the theory is to design the game rules to provide the desired behaviour as the optimal one.

Applied to scheduling in Desktop Grids, game-theoretic methods can serve several purposes. The literature considers the following two major examples. First, well-designed rules are able to provide optimal behaviour in a decentralized way; this can reduce the server load, eliminate necessity to gather information about the nodes, improve robustness of the schedule [15, 29, 36, 37]. Second, game-theoretic methods efficiently serve to counteract sabotage in volunteer computing Desktop Grid [6, 22, 56]. Besides the possibility to force saboteurs to stop their malicious activity, it is vital to understand interaction of intelligent and cooperating adversaries. However, the potential of game-theoretic methods seems to be much higher.

7 Enterprise Desktop Grids

Desktop Grids can be classified to volunteer computing projects, where volunteers grant their resources and are connected via the Internet, and Enterprise Desktop Grid systems that join resources of an institution and exploit local area networks. Much attention has been paid to task scheduling for volunteer computing: it is enough to mention the thesis [48] and a review [17]. Enterprise Desktop Grids are studied much less [26, 30, 31, 47]. However, these two types of Desktop Grids are quite different, also from the point of view of scheduling. Firstly, trust is much higher in Enterprise Desktop Grids: at least there are no saboteurs. Reliability is also higher. Secondly, nodes' behaviour is much more predictable [33, 46] and all information about them (including their performance, amount of memory, type of OS, installed software, etc) is available. In particular, availability patterns (distribution of ON and OFF periods) is more predictable. Thirdly, the structure of the Enterprise Desktop Grid is usually known. So, quite efficient task scheduling is possible with lower level of replication.

Next, Enterprise Desktop Grids are often relatively small in size, so that methods inapplicable in the general case can be used (e.g., in [5] linear optimization problem maximizes the throughput of a heterogeneous computing system).

Also, Enterprise Desktop Grid can be of different architecture, not necessarily adhere to just the classical client-server model. Peer-to-peer, torrent-like, multi-level computing networks can be used for solving various types of scientific problems. For example, in [13] a peer-to-peer torrent-like Desktop Grid where each node needs to get all the results is considered. One of the nodes serves as a tracker.

Finally, Enterprise Desktop Grid can exploit the PUSH model of interaction with nodes instead of the regular PULL model. Using the PUSH model, the server assigns a task to a node and is able to check the status of the task (completion progress), the status of the node (ON/OFF, available resources, availability periods, etc.); to cancel the task when the needed number of results is achieved. All this makes Enterprise Desktop Grids much closer to Computational Grid than to volunteer computing. But Enterprise Desktop Grids also keep many of the intrinsic characteristics of regular Desktop Grids.

Enterprise Desktop Grids allow to solve local scientific problems (e.g., data analysis) within an institution. However, their efficient usage needs new mathematical models and algorithms of task scheduling. The open problems of task scheduling in Enterprise Desktop Grids include both optimal replication, availability/reliability ratings, etc. (more important for volunteer computing) and special problems like “tail” computation or task grouping.

8 “Tail” computation

Another interesting problem is optimization of the “tail” computation. A distributed computational experiment involving a batch of tasks inherently consists of two stages ([8], see Fig. 1). At the first one, the throughput stage, the number of tasks is greater than the number of computing nodes (in the very beginning it can be much more). At this stage, the computational power is limiting the performance, so it is reasonable to supply each node with a task. With time, the number of unprocessed tasks decreases until it is equal to the number of computing nodes: then the second stage called the tail starts. At this stage, there is excess of computational power. In particular, during the first stage, the replication can be useless (from the point of view of the makespan) as it was shown in [25]. The same is true for parallel multicore implementations of the used algorithms. Tail computation is optimized in [8, 31].

A number of research problems related to specifics of Desktop Grids are connected to the two-staged batch completion.

The first problem is time estimation of the first stage completion. It is important for planning of the computa-

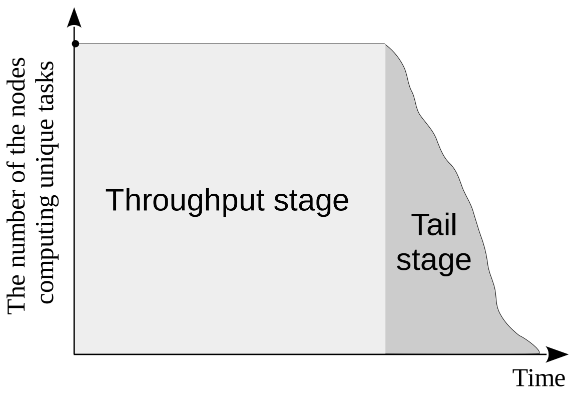

Figure 1: Two stages of batch completion in a Desktop Grid.

tional experiment, Desktop Grid maintenance, and keeping interest to the project among volunteers. This problem is complicated due to the dynamic nature of Desktop Grids: computational nodes can leave or join the computational network; high heterogeneity of computing nodes: computing performance of new or leaving nodes can significantly differ from the mean node performance; a computing network performance can have a trend (because of growth or lose an interest to the project among volunteers) or even flashes (as a result of “competitions” among volunteer teams).

Another problem is the fastest experiment or batch completion. In practice, “tail” computation takes a long time (usually, about two or three times greater than deadline) because of missing deadlines. A computational network does not have information of the current status of tasks computation. So, the “tail” accumulates a lot of nodes that have abandoned the computing network. As a certain task assigned to such a node violates the deadline, it is assigned again to a different node, possibly also going to leave the network soon. So, this prolongs the “tail”.

A solution to the problem is in the redundant computing: currently processing tasks are assigned to vacant computing nodes. This strategy significantly increases the chances that at least one copy is solved in time. Employing this strategy, one should to take into account characteristics of computational nodes (availability, reliability, computing power), processing the same task; accumulated task processing time; expected task completion time; and so on.

An approach for reducing the total batch completion time by about 14% and the “tail” phase, in particular, by about 40% is proposed in [53]: the nodes and the tasks are grouped by their dynamically evaluated performance and complexity.

Three strategies of task replication with performance of the nodes taken into account are experimentally tested in [31]: idle resources need to be used to decrease the makespan.

The fastest batch completion problem is even more complicated if a new tasks batch should be started after the current batch completion. In this case redundant computing reduces accessible computing power. The more complicated case is connected to inter-dependency between the tasks of the new batch and completion of certain tasks of the current batch.

9 Black box schedule generators

Attention to artificial intelligence (machine learning, neural networks) methods has been growing recently. They can be used for producing enhanced schedules. For example, in [44] neural networks forecast the load in order to schedule jobs to volunteer resources. The authors of [59] also apply neural networks to evaluate the load. Such systems have proved to be able to solve complex tasks of estimating fuzzy concepts like reliability, availability patterns, task complexity, etc.

Another approach is the genetic algorithms. Applied to scheduling (e.g., [18, 19, 41, 55]), they simulate the Darwinian evolution in a population of schedules to produce the best ones. Some authors use genetic algorithms to develop optimal scheduling policies.

10 Discussion

Scheduling in Desktop Grids is generally a multi-criteria optimization problem. The demands can be classified to:

- Performance or efficiency, measured as makespan, throughput, turnaround time, total time for the bath of tasks, overhead time, etc; early production of valuable results also belongs here.
- Reliability or trust, meaning low risk of getting a wrong result or high chances to get any result, including counter-sabotage measures.
- Justice, important mostly in volunteer computing, meaning taking into account both the desires of volunteers and the project goals (possibly, multiple projects).

Obviously, the criteria often contradict each other. Even criteria from the same class can be mutually exclusive; e.g., task replication can be used to get an answer sooner (performance) but on the cost of redundant use of resources. Also the criteria are quite hard to formalize. Even makespan (the time to complete a set of tasks) and throughput (tasks per hour) are not exactly reciprocals.

Black box algorithms, including machine learning, genetic search, and other, are able to solve such “fuzzy” problems. They are potentially able to improve efficiency, reliability, and justice, in any combination. The same is true for multi-agent models (game-theoretic models): first, decentralization definitely improves scalability, reduces the server load, forces the nodes to work better and thus is able to optimize efficiency. Also, well-chosen game rules can stop nodes from lying or poorly solve tasks, so that reliability is better (including an important question of stopping sabotage). Finally, game-style interaction of independent agents helps justice; e.g., nodes can distribute tasks of a few projects in a fair way.

As for the task grouping, it is generally promising for better efficiency, though, as we have noted, can be used for cross-checking. Creating parcels of variable size can be useful for fair use of computing power of the nodes.

Other problems that we discuss in the paper are more specific. Tail-stage optimization is mostly from the point of view of efficiency. Also, when serving multiple projects in a queue, resource management at the final stage becomes non-trivial.

Enterprise Desktop Grids are free of some volunteer computing problems, as has been discussed already. Therefore, the main criteria are those of the first class, i.e., efficiency. However, relatively low computing power in case of a few projects can be a challenge from the point of view of fair scheduling of the available resources. Also interests of computer users, project owners, and other concerned parties may not coincide. So, the fair distribution of the available resources without violating anybody’s rights can be a challenging problem.

Domain-specific optimization or optimal choice of the order of the search is mostly for efficiency in the special sense of getting results sooner. However, it is quite a new subject, so new applications can appear. Also (at the moment) optimizing the cost of the computing seems to serve only reliability. Though, we are sure that this approach can be useful also for the efficiency, including fewer deadline violations (if one is punished by a virtual penalty increasing the cost). Possibility to join the cost metrics with game-theoretic approach is (up to our knowledge) completely unstudied.

As for the “needle in a haystack” search, it is proposed for less server load and thus better efficiency; however, rare valuable answers are expensive, so this becomes a question of reliability.

Table 1: Effect of the discussed subjects.

A problem	Performance	Reliability	Justice
Task grouping	✓	✓	✓
Needle in a haystack	✓	✓	
Task cost	?	✓	?
Domain-specifics	✓	?	?
Game theory	✓	✓	✓
EDG	✓		✓
Tail stage	✓		✓
Black box	✓	✓	✓

So, most open problems selected by us serve at least two optimization purposes (see Table 1), which agrees well with a nature of heterogeneous computing systems where multiple conflicting interests meet and need to be resolved.

Conclusion

Task scheduling plays crucial role in Desktop Grids. Much efforts were taken to develop new mathematical models and algorithms. With some prerequisites they promise significant increase of computing performance. But there are also open problems in this domain.

We have considered several promising directions to improve task scheduling in Desktop Grids. They include task grouping (parceling), “needle in a haystack”, mean cost evaluation, domain-imposed scheduling objectives, game-theoretical methods of robust or decentralized scheduling, Enterprise Desktop Grids, optimizing the final part of the project, and sophisticated algorithms.

Solution of each of the described problems promises higher efficiency of Desktop Grids. Optimal task grouping reduces the server load, increasing the number of simultaneously served computing nodes and thus the Desktop Grid performance. The same is true for “needle in a haystack” problems, using some game-theoretical solutions, and the mean-cost approach. Domain-imposed criteria, mathematical models, and algorithms promise faster solutions of higher quality. Games of scheduling potentially give more effective protocols of interaction with computing nodes and more effective tasks distribution. Optimizing the “tail” computation reduces makespan. AI schedule generators are promising in producing efficient schedules that suit best for the given environment and take into account all available informtion. Finally, Enterprise Desktop Grids are the special type of computing systems not widely propagated. We believe that the described problems are hot topics of task scheduling in Desktop Grids.

Acknowledgement: The authors are grateful to the anonymous reviewers for their useful comments that helped to improve the paper.

The work is supported by the Russian Foundation for Basic Research (grant numbers 16-07-00622, 15-07-02354, and 15-29-07974).

References

[1] BoincIntro. URL: http://boinc.berkeley.edu/trac/wiki/BoincIntro, accessed Jun 2017.

[2] BOINCstats. URL: https://boincstats.com, accessed Jun 2017.

[3] DevProjects. URL: http://boinc.berkeley.edu/trac/wiki/DevProjects, accessed Jun 2017.

[4] JobIn. URL: http://boinc.berkeley.edu/trac/wiki/JobIn, accessed Jun 2017.

[5] I. Al-Azzoni and D.G. Down. Dynamic scheduling for heterogeneous desktop grids. *Journal of Parallel and Distributed Computing*, 70(12):1231–1240, dec 2010.

[6] A.F. Anta, Ch. Georgiou, M.A. Mosteiro, and D. Pareja. Multi-round master-worker computing: a repeated game approach. In *Reliable Distributed Systems (SRDS), 2016 IEEE 35th Symposium on*, pages 31–40. IEEE, 2016.

[7] A.L. Bazinet and M.P. Cummings. Subdividing long-running, variable-length analyses into short, fixed-length BOINC workunits. *Journal of Grid Computing*, sep 2015.

[8] Orna Agmon Ben-Yehuda, Assaf Schuster, Artyom Sharov, Mark Silberstein, and Alexandru Iosup. Expert: Pareto-efficient task replication on grids and a cloud. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 167–178. IEEE, 2012.

[9] J Celaya and U Arronategui. A task routing approach to large-scale scheduling. *Future Generation Computer Systems*, 29(5):1097–1111, 2013.

[10] I. Chernov. Theoretical study of replication in desktop grid computing: Minimizing the mean cost. In *Proceedings of the 2nd Applications in Information Technology (ICAIT-2016), International Conference on*, pages 125–129, Aizu-Wakamatsu, Japan, 2016.

[11] I.A. Chernov, E.E. Ivashko, and N.N. Nikitina. Survey of task scheduling methods in Desktop Grids. *Program Systems: Theory and Applications*, 8(3):3–29, 2017. In Russian.

[12] I.A. Chernov and N.N. Nikitina. Virtual screening in a desktop grid: Replication and the optimal quorum. In V. Malyshkin, editor, *Parallel Computing Technologies, International Conference on*, volume 9251, pages 258–267. Springer, 2015.

[13] G. Chmaj, K. Walkowiak, M. Tarnawski, and M. Kucharzak. Heuristic algorithms for optimization of task allocation and result distribution in peer-to-peer computing systems. *International Journal of Applied Mathematics and Computer Science*, 22(3):733–748, 2012.

[14] S.J. Choi, H.S. Kim, E.J. Byun, and C.S. Hwan. A taxonomy of desktop grid systems focusing on scheduling. Technical report KU-CSE-2006-1120-02, Department of Computer Science

and Engeering, Korea University, 2006.

[15] B. Donassolo, A. Legrand, and C. Geyer. Non-cooperative scheduling considered harmful in collaborative volunteer computing environments. In *Cluster, Cloud and Grid Computing, 11th IEEE/ACM International Symposium on*, pages 144–153, 2011.

[16] W. Du, J. Jia, M. Mangal, and M. Murugesan. Uncheatable grid computing. *Electrical Engineering and Computer Science*, Paper 26:1–8, 2004.

[17] N.M. Durrani and J.A. Shamsi. Volunteer computing: requirements, challenges, and solutions. *Journal of Network and Computer Applications*, 39:369–380, mar 2014.

[18] T. Estrada, O. Fuentes, and M. Taufer. A distributed evolutionary method to design scheduling policies for volunteer computing. *ACM SIGMETRICS Performance Evaluation Review*, 36(3):40–49, 2008.

[19] T. Estrada and M. Taufer. Challenges in designing scheduling policies in volunteer computing. In C. Cérin and G. Fedak, editors, *Desktop Grid Computing*, pages 167–190. CRC Press, 2012.

[20] Z. Farkas and P. Kacsuk. Evaluation of hierarchical desktop grid scheduling algorithms. *Future Generation Computer Systems*, 28(6):871–880, jun 2012.

[21] Z. Farkas, A. Marosi, and P. Kacsuk. Job scheduling in hierarchical desktop grids. In F. Davoli, N. Meyer, R. Pugliese, and S. Zappatore, editors, *Remote Instrumentation and Virtual Laboratories*, pages 79–97. Springer, Boston, MA, 2010.

[22] A.A. Fernández, C. Georgiou, M.A. Mosteiro, and D. Pareja. Algorithmic mechanisms for reliable crowdsourcing computation under collusion. *PLoS ONE*, 10(3):1–22, 2015.

[23] Stefano Forli, Ruth Huey, Michael E Pique, Michel F Sanner, David S Goodsell, and Arthur J Olson. Computational protein-ligand docking and virtual drug screening with the autodock suite. *Nature protocols*, 11(5):905–919, 2016.

[24] I.E. Gabis and I.A. Chernov. *The Kinetics of Binary Metal Hydride Decomposition*. Chemistry Research and Applications. Nova Publisher, 2017.

[25] Gaurav D Ghare and Scott T Leutenegger. Improving speedup and response times by replicating parallel programs on a SNOW. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 264–287. Springer, 2004.

[26] D.L. González, G.G. Gil, F.F. de Vega, and B. Segal. Centralized BOINC resources manager for institutional networks. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8. IEEE, 2008.

[27] P. Kacsuk, J. Kovacs, Z. Farkas, et al. SZTAKI desktop grid (SZDG): A flexible and scalable desktop grid system. *Journal of Grid Computing*, 7(4):439, 2009.

[28] M.Kh. Khan, T. Mahmood, and S.I. Hyder. Scheduling in desktop grid systems: Theoretical evaluation of policies and frameworks. *International Journal of Advanced Computer Science and Applications*, 8(1):119–127, 2017.

[29] J. Kołodziej and F. Xhafa. Meeting security and user behavior requirements in grid scheduling. *Simulation Modelling Practice and Theory*, 19(1):213–226, jan 2011.

[30] D. Kondo and H. Casanova. Computing the optimal makespan for jobs with identical and independent tasks scheduled on volatile hosts. Technical report CS2004-0796, Dept. of Computer Science and Engineering, University of California, San Diego, 2004.

[31] D. Kondo, A.A. Chien, and H. Casanova. Scheduling task parallel applications for rapid turnaround on enterprise desktop grids. *Journal of Grid Computing*, 5(4):379–405, oct 2007.

[32] Ilya Kurochkin and Anatoliy Saevskiy. BOINC forks, issues and directions of development. *Procedia Computer Science*, 101:369–378, 2016. 5th International Young Scientist Conference on Computational Science, YSC 2016, 26-28 October 2016, Krakow, Poland.

[33] J.L. Lerida, F. Solsona, P. Hernandez, F. Gine, M. Hanzich, and J. Conde. State-based predictions with self-correction on Enterprise Desktop Grid environments. *Journal of Parallel and Distributed Computing*, 73(6):777–789, 2013.

[34] Chunlin Li and Layuan Li. Utility-based scheduling for grid computing under constraints of energy budget and deadline. *Computer Standards & Interfaces*, 31:1131–1142, 2009.

[35] M. Maheswaran, Sh. Ali, H.J. Siegel, D. Hensgen, and R.F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. *Journal of Parallel and Distributed Computing*, 59(2):107–131, 1999.

[36] V.V. Mazalov, N.N. Nikitina, and E.E. Ivashko. Hierarchical two-level game model for tasks scheduling in a desktop grid. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2014 6th International Congress on*, pages 541–545. IEEE, 2014.

[37] V.V. Mazalov, N.N. Nikitina, and E.E Ivashko. Task scheduling in a desktop grid to minimize the server load. In V. Malyshkin, editor, *Parallel Computing Technologies, International Conference on*, volume 9251, pages 273–278. Springer, 2015.

[38] S. Nesmachnow, B. Dorronsoro, J.E. Pecero, and P. Bouvry. Energy-aware scheduling on multicore heterogeneous grid computing systems. *Journal of Grid Computing*, 11:653–680, 2013.

[39] Natalia Nikitina, Evgeny Ivashko, and Andrei Tchernykh. Congestion game scheduling implementation for high-throughput virtual drug screening using boinc-based desktop grid. In *International Conference on Parallel Computing Technologies*, pages 480–491. Springer, 2017.

[40] A.-C. Orgerie, L. Lefévre, and J.-P. Gelas. Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. In *Parallel and Distributed Systems, 14th IEEE International Conference on*, pages 171–178. IEEE, 2008.

[41] B. Qu, Y. Lei, and Y. Zhao. A new genetic algorithm based scheduling for volunteer computing. In *Computer and Communication Technologies in Agriculture Engineering (CCTAE), 2010 International Conference On*, volume 3, pages 228–231. IEEE, 2010.

[42] Rob E Quick, Samy Meroueh, Soichi Hayashi, Rynge Mats, Scott Teige, David Xu, and Bo Wang. Building a chemical-protein interactome on the open science grid. In *International Symposium on Grids and Clouds (ISGC) 2015, 15–20 March 2015, Academia Sinica, Taipei, Taiwan*, pages 1–5, 2015.

[43] Josep Rius, Fernando Cores, and Francesc Solsona. Cooperative scheduling mechanism for large-scale peer-to-peer computing systems. *Journal of Network and Computer Applications*, 36(6):1620 – 1631, 2013.

[44] Saddaf Rubab, Mohd Fadzil Hassan, Ahmad Kamil Mahmood, and Syed Nasir Mehmood Shah. Proactive job scheduling and migration using artificial neural networks for volunteer grid. In *Computer Science and Engineering, First EAI International Conference on*. EAI, 3 2017.

[45] Chetan Rupakheti, Aaron Virshup, Weitao Yang, and David N Beratan. Strategy to discover diverse optimal molecules in the small molecule universe. *Journal of chemical information and*

modeling, 55(3):529–537, 2015.

[46] S.A. Salinas. PFS: A productivity forecasting system for desktop computers to improve grid applications performance in Enterprise Desktop Grid. *Computing and Informatics*, 33:783–809, 2014.

[47] S.A. Salinas, C.G. Garino, and A. Zunino. An architecture for resource behavior prediction to improve scheduling systems performance on enterprise desktop grids. In *Advances in New Technologies, Interactive Interfaces and Communicability*, pages 186–196. Springer, 2012.

[48] L.F. Sarmenta. *Volunteer computing*. PhD thesis, Massachusetts Institute of Technology, 2001.

[49] S.S. Sathya and K.S. Babu. Survey of fault tolerant techniques for grid. *Computer Science Review*, 4:101–120, 2010.

[50] D. Szajda, B. Lawson, and J. Owen. Hardening functions for large-scale distributed computations. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, page 7946298. IEEE, 2003.

[51] A. Tchernykh, J.E. Pecero, A. Barrondo, and E. Schaeffer. Adaptive energy efficient scheduling in peer-to-peer desktop grids. *Future Generation Computer Systems*, 36:209–220, 2014.

[52] D. Toth and D. Finkel. Improving the productivity of volunteer computing by using the most effective task retrieval policies. *Journal of Grid Computing*, 7(4):519–535, dec 2009.

[53] M. Ujhelyi, P. Lacko, and A. Paulovic. Task scheduling in distributed volunteer computing systems. In *Intelligent Systems and Informatics (SISY), 2014 IEEE 12th International Symposium on*, pages 111–114. IEEE, 2014.

[54] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 2007.

[55] X. Wang, Ch.Sh. Yeo, R. Buyya, and J. Su. Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm. *Future Generation Computer Systems*, 27(8):1124–1134, oct 2011.

[56] Y. Wang, J. Wei, Sh. Ren, and Yu. Shen. Toward integrity assurance of outsourced computing — a game theoretic perspective. *Future Generation Computer Systems*, 55:87–100, 2016.

[57] F. Xhafa and A. Abraham. Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, 26(4):608–621, apr 2010.

[58] Jianhua Yu, Yue Luo, and Xueli Wang. Deceptive detection and security reinforcement in grid computing. In *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*, pages 146–152. IEEE, 2013.

[59] K.-M. Yu, Z.-J. Luo, C.-H. Chou, C.-K. Chen, and J. Zhou. A fuzzy neural network based scheduling algorithm for job assignment on computational grids. In T. Enokido, L. Barolli, and M. Takizawa, editors, *Network-Based Information Systems*, volume 4658, pages 533–542, Berlin, Heidelberg, 2007. Springer.