A Survey of Desktop Grid Scheduling

Evgeny Ivashko, Ilya Chernov, and Natalia Nikitina

Abstract—The paper surveys the state of the art of task scheduling in Desktop Grid computing systems. We describe the general architecture of a Desktop Grid system and the computing model adopted by the BOINC middleware. We summarize research papers to bring together and examine the optimization criteria and methods proposed by researchers so far for improving Desktop Grid task scheduling (assigning tasks to computing nodes by a server). In addition, we review related papers, which address non-regular architectures, like hierarchical and peer-to-peer Desktop Grids, as well as Desktop Grids designed for solving interdependent tasks.

Index Terms—Scheduling and task partitioning, scheduling

1 Introduction

Desktop Grid (DG) is a promising high-throughput computing paradigm. It is a form of a distributed computing system that uses idle time of non-dedicated geographically distributed computing nodes connected over a low-speed network. This idea is believed to have been first described by Litzkow et al. in 1987 [1]. Later on, in 1999, Sarmenta and Hirano introduced the term "volunteer computing" (VC) [2] to describe the possibility of almost unlimited scaling the computational resources constituting DG by means of VC power. In 2001, Foster et al. [3] made a clear distinction between Computational Grids, Enterprise DGs, and VC; here we follow this distinction.

As a practical tool, DG traces its history back to 1996, when the first large VC project GIMPS [4] was launched, followed by another large project *distributed.net* [5] and the introduction of SETI@home [6] in 1997. Later, in 1999, SETI@home and Folding@home [7] were launched as the first public VC scientific projects. Due to the success of these and many other projects (such as ClimatePrediction.net, Rosetta@home, Docking@home, Einstein@Home, the IBM World Community Grid etc., to name a few), DGs have received growing interest and attraction. In 2002 the group that developed SETI@home started the Berkeley Open Infrastructure for Network Computing (BOINC) project to develop a general-purpose open-source VC middleware [8].

The current growing potential of DGs is connected with growing performance of desktop and laptop computers, their increasing number, faster network connections, cheaper internet traffic, higher availability of wireless networks, etc.

- *The authors are with the Institute of Applied Mathematical Research, Karelian Research Center, Russian Academy of Sciences, Petrozavodsk 119991, Russia. E-mail: {ivashko, nikitina}@krc.karelia.ru, IAChernov@yandex.ru.*

Manuscript received 15 Nov. 2017; revised 9 June 2018; accepted 19 June 2018. Date of publication 25 June 2018; date of current version 9 Nov. 2018. (Corresponding author: Ilya A. Chernov.)
Recommended for acceptance by A. Benoit.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2850004

To date, there are many middleware systems for Desktop Grid computing. Even restricting oneself with centralized systems of the "master-worker" paradigm, one names more than ten platforms (BOINC [8], the original SETI@home [6], XtremWeb [9], Cosm [10], X-Com [11], Entropia [12], Bayanihan [2], Javelin [13], HTCondor [14], Grid MP [15], Alchemi [16] etc.). Decentralized, peer-to-peer and other variations make this list even longer. However, the BOINC platform has been developed with the purpose of simplifying and unifying creation and operation of DG projects both at the enterprise or the Internet level. For the moment, BOINC has succeeded in becoming a de-facto standard and the most widely used middleware for DGs and VC.

This is the reason we survey task scheduling in DGs that inherit the key design features presented in BOINC:

- centralized architecture;
- master-worker computing paradigm;
- PULL work assignment mode.

Today DGs constitute an important part of high-performance computing landscape along with Computational Grid systems, computing clusters, and supercomputers. However, as opposed to these tools, the DG computing paradigm is not so thoroughly studied. The experience of the multitude of successful BOINC projects [17] has uncovered new concerns and problems unique to DGs.

Task scheduling plays a crucial role in DG-based high-throughput computing. It can drastically affect the performance of a DG system. In comparison with Computational Grids and computing clusters, task scheduling in DGs is more complicated because of such factors as hardware and software heterogeneity, lack of trust, uncertainty on availability and reliability of computing nodes, etc.

A wide range of algorithms and heuristics has been proposed in the literature to address task scheduling challenges in DGs. Additional information is gathered and used: availability periods, reliability ratings, and so on. The aim of this work is to review research papers devoted to optimizing performance of DGs by scheduling. We bring together and examine the optimization criteria and methods proposed by

researchers so far. Alongside the conventional DGs, special cases of them are considered in the literature: hierarchical, peer-to-peer, etc. We also briefly review some related papers, which address these non-regular architectures.

There are several surveys in the domain of DGs (including the task scheduling problem) that have been presented by now. They review task scheduling taxonomies [18], heuristic methods [19], and knowledge-based algorithms [20], [21], popular middleware systems and the factors influencing their performance [22]. We discuss these works in more detail in Section 3.3.

In contrast to these surveys, our research is

- concentrated on task scheduling optimization criteria and methods;
- comprehensive, as it covers the great majority of research papers in the domain of DG task scheduling;
- up-to-date, as it covers papers up to 2017.

This work provides a survey of DG task scheduling research. We consider global scheduling policies describing the rules of assignment of tasks to computing nodes by a server. We have studied 127 research papers dated from 1999 to 2017 and related to DG task scheduling.

The rest of the paper is organized as follows. Section 2 highlights specifics of the DG concept and BOINC as the most commonly used middleware for DGs. Section 3 gives an analysis of BOINC task scheduling research papers. We describe our research objects, overview the problems, optimization criteria, methods, and results achieved by researchers.

2 Desktop Grid and BOINC

2.1 Desktop Grid and the BOINC Platform

DG is a form of distributed high-throughput computing system that uses idle time of non-dedicated geographically distributed computing nodes connected over a low-speed network (by "low-speed" we mean slow compared to supercomputer interconnect).

DGs are of special interest because of their advantages, including the following:

- ease of deployment and support,
- low cost,
- high scalability (linear for suitable problems),
- significant potential peak performance (10^6 factor of a common PC),
- ease of software development,
- possibility to use legacy software.

But there are also several features that scale down the class of problems efficiently solved by DGs and pose implementation challenges. They are:

- slow connection;
- limited computational capacity of separate nodes;
- heterogeneity of software, hardware, or both;
- lack of availability information;
- low reliability of computing nodes;
- lack of trust to computing nodes.

These features are the source of internal uncertainties present in DG systems. The uncertainties decrease the system performance, the completion time of tasks or batches, availability and reliability of nodes, etc. This leads to significant decrease in effective computing performance (the number of unique tasks solved in a time unit) compared with the peak performance, sometimes by tens of times.

Computational clusters and Computational Grids have dedicated computing nodes, while DGs do not; this affects nodes' availability, reliability, and trust. DGs usually have a large number of computing nodes, much larger than Computational clusters or Computational Grids (for the latter we count the constituting high-performance computing systems as computing nodes and do not address their internal structure). This leads to high heterogeneity. Using idle time of different computing devices of diverse performance, with various hardware, operational systems, software libraries, etc., increases heterogeneity even more. Also, the set of computing nodes is changing in time. A DG can have one or two scheduling levels: the latter happens when a computing node shares its resources among several DG applications. Most of these characteristics also separate DGs from cloud computing systems (see, e.g., [23]). However, DGs can be joined with computing clusters [24], Grid systems [25], or cloud computing systems [26] to establish more complex and powerful computational platform.

All these characteristics illustrate high scheduling complexity of DGs in comparison with two other conventional computational paradigms, computational clusters and Computational Grids.

Scheduling-related scientific works consider different (not mutually exclusive) types of a DG:

- *VC system*: computing nodes are personal computers of volunteers connected over the Internet. The scheduler faces heterogeneity, low network speed connection, unreliability, etc.
- *Enterprise DG*: computing nodes are workstations of an organization connected over a local area network. There is an option to assign tasks to specific computing nodes to improve scheduling characteristics.
- *hierarchical DG*: there is an hierarchy of servers; each subordinate server requests batches of tasks from its master for its own computing nodes. The scheduler should be aware of the hierarchy and take into account the tasks redistribution.
- *peer-to-peer system*: computing nodes can communicate with each other. The scheduler should be aware of the communication graph (including information on the connection speed); tasks can be redistributed by computing nodes aiming to decrease the server load or increase the data transfer speed.
- *solving dependent tasks*: tasks interdependencies are described as a directed acyclic graph (DAG), and a task cannot be solved until all antecedent tasks it depends on are solved. Dependence severely complicates scheduling. Note that a set of independent tasks of the same origin is often called *bag of tasks* (BoT).

2.2 BOINC Architecture and Native Scheduling

There are a number of middleware systems for DG computing. However, the open source BOINC platform [8] is nowadays considered as a de facto standard among them. BOINC

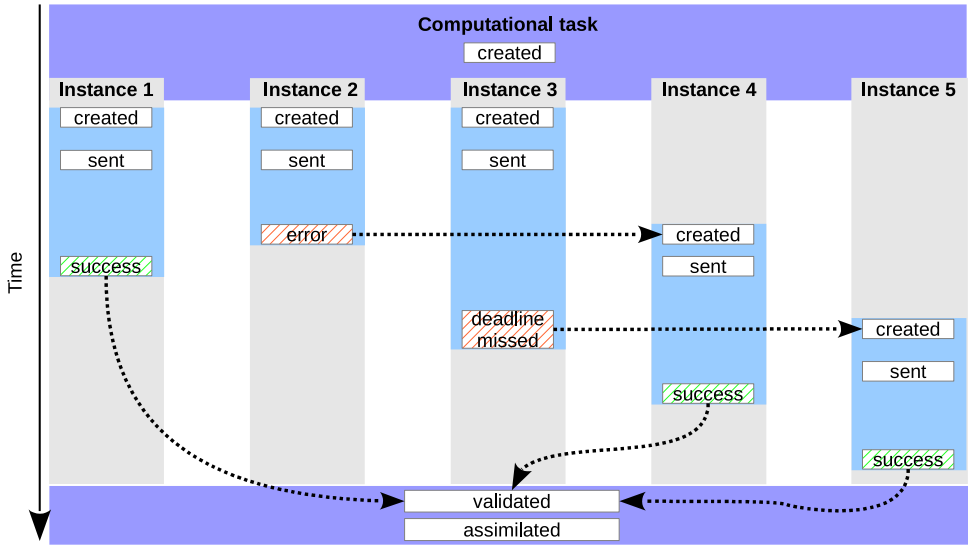

Fig. 1. Example of the lifetime of a computational task in BOINC.

has been a framework for many independent VC projects since the early 2,000 s. After the launch of SETI@home, many other VC projects have gained popularity. They fall into four types:

- a stand-alone project dedicated to a single scientific problem (Asteroids@home, ClimatePrediction.net, Einstein@Home, SAT@home etc.);
- a project involving several different applications on the same problem (LHC@home, GPUGrid.net etc.);
- an umbrella project: several different problems united by, e.g., their general subject (the IBM's World Community Grid etc.), or the national affiliation (CAS@home of China, Ibercivis of Spain etc.);
- a private project that serves the needs of a local scientific group.

In total, about 60 BOINC projects are active and publicly available today, comprising more than 1.5 million computing nodes [17].

BOINC uses the client-server architecture. The client part is able to employ idle resources of a computer (of various hardware and software) for computations within BOINC projects, with minimal action from the computer's owner.

The server part consists of several subsystems responsible for tasks generation, distribution, results reception, assimilation, etc.

The computing model employed in BOINC implies that each computational task can exist in multiple *replicas*. Each is computed separately and their results are compared. Technically, a result may be identified as *valid* or not. First, the validation process verifies that all files associated with the result have been present to the server and have the right format. Second, in case of replication, the result is identified as valid if it coincides with a majority of received results; other ones are identified as not valid.

Note that a valid result may be wrong from the point of view of the solved scientific problem. Verification may be automated or be complex and manual. Table 2 in Section 3 illustrates that some authors distinguish valid and correct results [27], [28], [29], [30], [31], [32], while some do not [33], [34], [35], [36], [37].

The *quorum* concept is used to define the minimal number of successful results to obtain for one computational task. The BOINC settings allow creating and distributing more task replicas dynamically as needed. An individual *deadline* is set for each task instance to limit its completion time. If the server does not get a result before the deadline, the task instance is considered lost.

In Fig. 1 we exemplify the lifetime cycle of a computational task in BOINC. In this example, the quorum is set equal to three. Initially, three instances of a task are created and sent to the clients. As soon as the second of them is finished with an error, the server creates one more instance (otherwise the quorum of three would never be reached). Meanwhile, the third instance gets lost and misses its deadline. The server gives up on that instance and creates the fifth one. It is successfully computed, together with the first one and the fourth one, so the quorum is reached. The computational task gets validated and assimilated.

The replication mechanism as a form of redundant computing can serve a number of purposes, first of all, the reliability, by increasing the chance to obtain the correct answer in time even if some nodes switch off without having finished the task. This helps, in its turn, to improve efficiency, provided that nodes are unreliable in general, by decreasing average time of waiting for an answer.

Reduction of the deadline violation risk is even more important when solving interdependent tasks. Replication and voting are important in VC as a counter-sabotage defense measure. The reputational quorum is a method of voting with higher-ranked nodes' votes valued more: this approach entails both replication and reputation techniques at once.

The built-in BOINC scheduling mechanism consists of the server scheduling part needed to assign tasks to computing nodes according to the global scheduling policies, and the local scheduling part needed to manage the task execution on the nodes so as to meet deadlines, to fairly share the client resources among several BOINC projects, and so on.

In addition, the mechanisms of *locality scheduling* and *homogeneous redundancy* can be employed. The locality scheduling is aimed at minimizing the amount of data transferred to clients. The tasks that require large input files

are being preferentially dispatched to the hosts that already have them. The homogeneous redundancy mechanism was implemented to deal with numerical discrepancies that arise in computational results due to different client characteristics. The clients are clustered according to their OS and CPU architecture, and all replicas of a given task are dispatched only to clients within the same class.

Today, the BOINC scheduling model has a rich basis that has been developed for more than 10 years. To name a few of the implemented features [38], it includes:

- a mechanism of adaptive replication to minimize replication overheads using highly reliable hosts;
- fair resource allocation mechanism among multiple job submitters in a single BOINC project;
- a mechanism of runtime estimation based on historical statistics either per host or per app version;
- a mechanism to set multiple pre-defined classes of jobs varying by their size, so that the scheduler will send jobs of appropriate size to a host depending on its capabilities;
- a score-based scheduling framework, which incorporates multiple goals and criteria.

Existing mechanisms provide flexible and powerful technical tools to adjust BOINC project performance on top of the default scheduler. Nevertheless, the overall impact of these mechanisms on the system performance of a BOINC project might have been studied better. Effective computational performance can be increased by sophisticated task scheduling algorithms that do not only integrate existing technical capabilities but also employ apriori-known additional information about the computing network structure, the computing nodes characteristics, the subject area specifics, etc. Their further implementation can be based either on the existing mechanisms such as the ones listed afore, or on the low-level altering of the server software, or integration of special additional programs.

One type of such additional information is job runtime estimation. Having node availability information and job runtime estimation, one can develop more effective schedulers. For example, the paper [39] describes an approach to scheduling a large number of jobs with heterogeneous requirements to grid resources, which include VC resources. The system is based on apriori runtime estimation using machine learning with random forests. Papers [40] and [41] propose a novel solution for scheduling parameter sweep jobs. It is based on runtime prediction used for mapping jobs onto available resources in order to reduce the number of jobs prematurely interrupted due to insufficient resource availability. Predictions are made by GIPSy (Grid Information Prediction System). In the paper [42] a dynamic replication approach is proposed to reduce the time needed to complete a batch of tasks. A mathematical model and a strategy of dynamic replication at the tail stage are developed. The model is based on job runtime estimation. DG emulators and simulators also can be used to estimate either jobs or batch of tasks runtime. BOINC platform itself has advanced capabilities on jobs runtime estimation [38].

There are several papers devoted to local (client-side) scheduling optimization. For example, Atlas et al. [43] use multi-criteria optimization. Based on the proposed algorithm, volunteer preferences are not necessarily completely satisfied, but the performance is better compared to the native BOINC algorithms, as it is shown by emulation on EmBOINC. Anglano and Canonico [44] also consider scheduling multiple BoTs at once, so a Bag should be chosen along with the usual scheduling tasks of this Bag. No information about tasks or nodes is used. The WQR-FT algorithm (WorkQueue with Replication Fault Tolerant [45], obtained from the WQR [46] algorithm by adding checkpointing and replication) is used for scheduling, while five heuristics for choosing a BoT are compared by simulation.

Testing scheduling algorithms on real DGs is quite difficult and costly. Therefore simulation or emulation techniques are often used. The former means running a model of the computational system; the latter means using a real server with simulated clients. A number of papers are devoted to these aspects.

Anderson [47] proposes a BOINC client emulator that allows trialing different scheduling strategies in various usage scenarios, namely with varying hardware characteristics, client availability patterns, errors in job runtime estimation, etc. The BOINC scheduling policies employed back then are described and evaluated according to several performance metrics. The results show which scheduling strategies are the most efficient in different conditions. Beaumont et al. [48] consider simulation of various computing systems focusing on the SimGrid simulator. A part of chapter [20] by Estrada and Taufer is devoted to reviewing and description of SimBA, SimBOINC, and EmBOINC software. Estrada et al. describe DG emulation by EmBOINC in detail in [49] and [50], also paying attention to simulation tools. Taufer et al. [34] describe the BOINC simulator SimBA. Alonso-Monsalve et al. [51] propose a complete simulator that takes into account the whole BOINC infrastructure: projects, servers, network, redundant computing, scheduling, and volunteer nodes. The simulator allows analyzing a wide range of characteristics: the throughput of each project, the number of jobs executed by the clients, the total credit granted, and the average occupation of the BOINC servers.

Now we analyse the research papers devoted to various aspects of global task scheduling in BOINC-based DGs.

3 Scheduling Papers Review

3.1 Sources Overview

In order to compose a systematic and comprehensive survey, we selected scholarly papers in the following way. First, the search using the following keywords was performed using the Scopus database:

- DG task/job/resource scheduling,
- volunteer computing task/job/resource scheduling,
- grid computing task/job/resource scheduling,
- BOINC task/job/resource scheduling.

These papers composed the major part of the list. Second, after searching the Web of Science and the Google Scholar databases by the same keywords, we expanded the list with several more papers. Third, the tables of contents of the leading world journals covering the subject of DG computing were searched by the same keywords.

The found articles were brought together in a spreadsheet. The general subject of each paper was manually extracted

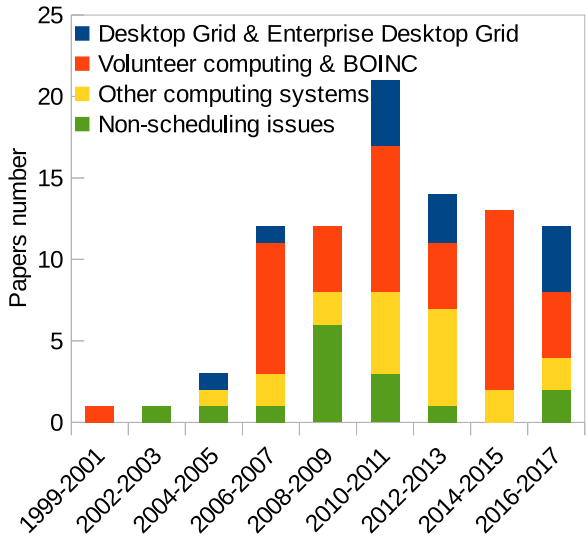

Fig. 2. Timeline of papers on task scheduling and related problems.

based on the abstract and brief examination of the text. As the survey concentrates on a specific DG type, 63 out of 127 papers were filtered out at this stage. Finally, the reference lists of the selected papers were manually scanned to add the most significant sources to the spreadsheet.

The survey comprises research works dated from 1999 to 2017. In Fig. 2 we present the distribution of selected papers over the time periods. By non-scheduling issues we mean counter-sabotage measures, availability prediction, presenting software (e.g., simulators and emulators), etc.

The surveyed works were published as regular papers in 31 different journals, contributions in 29 conference proceedings, one book chapter, one scientific report and one technical report. The distribution of papers over publication sources is presented in Fig. 3.

3.2 Optimization Criteria and Methods

The problem of scheduling is shown to be hard [19], [52], [53], [54] even if all information is available. However, in realistic DG setting it is even harder. First, the set of tasks is seldom available in advance and their complexities are unknown. Second, the set of computing nodes changes in time: the nodes can become unavailable, temporarily or forever; they can be unreliable; new nodes with unknown properties can join the DG at any time in case of VC.

One needs a numerical rating to compare scheduling schemes. The choice is not obvious, for there are multiple numerical characteristics of scheduling policies, sometimes in conflict with each other. For example, efficiency often contradicts reliability. Here is a list of the most popular optimization parameters applied in task scheduling.

- *Throughput* as the number of tasks completed or valid results obtained per a time unit [28], [31], [32], [33], [34], [35], [36], [37], [43], [55], [56].
- *Makespan* as the time interval to complete a set of computational tasks [27], [53], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70].
- *Turnaround time* of a computational task as the time from its creation to obtaining its result. [44], [70], [71].
- *Reliability* as the probability of the server to receive a correct result, a valid result, or any result once a task instance has been sent out [27], [28], [29], [30], [31], [32], [43], [55], [59], [66], [67], [72].
- *Availability* as the ratio of results returned to the server to the total number of tasks sent out to the node [30], [43], [55], [59], [66], [67], [72].
- *Load* of the computational server or computational nodes [47], [73], [74], [75].
- *Overhead* due to replication or failures [29], [32], [47], [56], [58], [59], [63], [64], [75], [76], [77], [78].
- *Cost of an answer*, including *energy consumption* [32], [79], [80], [81], [82], [83].

Beside the listed common parameters, more special ones are often considered. Among them are the sabotage tolerance, important mostly for VC; optimal processing of dependent tasks; serving several simultaneous projects; prediction of nodes availability; optimal handling of diverse nodes [32], [66], [75], [84], [85], [86], [87].

The main contribution of most researchers is to propose and validate a special method to optimize a criterion or a set of criteria. Different in details, all the optimization methods can be divided into the following types:

- *Grouping or clustering of the nodes* according to certain properties (e.g., reliability, capacity, availability) is often useful for efficient scheduling by itself or combined with other techniques. The group concept is used for voting, and the tasks are often being scheduled to certain groups to be then locally distributed within the group. Used in [28], [31], [36], [37], [53], [55], [59], [60], [61], [71], [74], [76], [88].

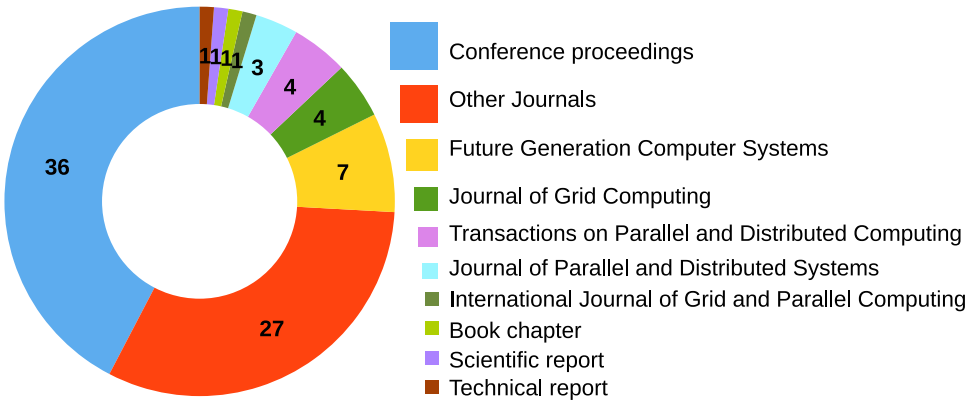

Fig. 3. Publication sources of the surveyed papers (1999–2017).

TABLE 1
Mathematical Methods Employed in Papers on DG Scheduling

Paper	Mathematical method
Chernov and Nikitina, 2015 [32] Yasuda et al., 2015 [31] Miyakoshi et al., 2014 [30] Rumiantsev, 2014 [60] Bouguerra et al., 2011 [63] Canon et al., 2011a [29] Watanabe et al., 2011 [28] Heien et al., 2009 [72] Byun et al., 2007 [67] Gao and Malewicz, 2007 [27] Sonnek et al., 2007 [37] Choi et al., 2004 [70]	Probabilistic analysis
Canon et al., 2014 [91] Canon et al., 2011b [64] Al-Azzoni and Down, 2010 [71] Fujimoto and Hagihara, 2006 [69] Sonnek et al., 2006 [36]	Discrete optimization
Kondo and Casanova, 2004 [68]	Continuous analysis
Mazalov et al., 2015 [73] Mazalov et al., 2014 [74] Nikitina et al., 2018 [92]	Game theory
Byun et al., 2007 [67] Bouguerra et al., 2011 [63] Smaoui and Garbey, 2013 [62]	Markov chains Markov decision process Monte Carlo functions

- *Reputation of the nodes or their groups* is estimated by the history of the work and can reflect either reliability or availability. This can be defined as the probability to present an answer before the deadline or to present the correct answer, or not to act maliciously, etc. Trust techniques are also reputation-based: blacklisting, recommendation, and other approaches help to improve scheduling efficiency and quality of obtained results. Used in [28], [30], [33], [34], [36], [37], [58], [77], [85].
- *Replication* is scheduling the instances of the same task to two or more nodes. It is expected to increase reliability by exposing saboteurs, revealing failing nodes, checking the answers, etc. Also, it is able to improve performance by fewer deadline violations, lower answer awaiting time, etc. A quorum is a fixed number of identical answers to be accepted as the truth. Voting is the selection of the correct answer by the majority of the voters, possibly of a group. Replication is used in [28], [30], [32], [36], [37], [42], [60], [72], [76], [89].
- Some heuristics, often based on the greedy scheme, are proposed and tested by *simulation* (running a model of DG) or *emulation* (running a DG with real server and simulated clients). Simulation is used more often: among the papers listed in Table 2, only two [20], [34] use emulation. Using heuristic algorithms and simulation is necessary because scheduling problems are often NP-hard [19], [52], [90].
- Some authors explore scheduling models analytically. In Table 1, there is a summary of mathematical methods employed in the surveyed papers.

Definitions of optimization criteria and applied methods are the basis for the development of a specific scheduling algorithm. In the next sections, we observe the papers considering optimization criteria related to the computational performance of DGs and the methods used to develop scheduling algorithms.

3.3 Survey Papers

Several research papers survey existing literature on task scheduling from different points of view, including a number of attempts on taxonomy and classification.

The paper of Choi et al. [18] provides several taxonomies of DGs. In particular, there is a scheduler's perspective taxonomy. Scheduling is considered from the points of view of the application (task dependency, data- or computing intensity, divisibility, etc.), the resource (dedication, trust, failure, heterogeneity), and the scheduler. The latter includes organization (centralized, hierarchical, or distributed scheduling), push or pull mode, scheduling and grouping policies, static, dynamic, or adaptive scheduling, fault tolerance, load balancing, and optimization criteria. Being relatively outdated (published in 2007), it does not take into account all optimization criteria used in newer papers. Although many existing projects are described and classified, the review lacks a description of practical techniques.

The review of Xhafa and Abraham [19] considers Computational Grids: they highlight scheduling complexities (like resources/jobs, heterogeneity, dynamic structure, and so on), define different types of schedulers, divide the scheduling procedure into several steps, define several types of scheduling in Grids. Considered scheduling methods include those dealing with interdependent tasks, centralized, hierarchical, and distributed, static and dynamic, individual and batch, adaptive, and data-oriented. The models of task scheduling are described from the points of view of information availability, optimization criteria, etc. The main results of the paper are the computational models for Grid scheduling and the analysis of several scheduling heuristics and meta-heuristics. Using heuristics is motivated by the fact that the considered problems are computationally hard. Though the paper is devoted to Computational Grids, it contributes a lot to scheduling types classification and scheduling methods development.

The chapter by Estrada and Taufer [20] reviews scheduling algorithms for DGs and the challenges they face. They are classified into three groups: naive that ignore the work history; knowledge-based that use information about the nodes, including accumulated statistics; and adaptive that change their parameters on the fly. Several algorithms from each group are discussed in detail, including an original adaptive genetic algorithm similar to the one from [56] (see the next section). Also, the simulators SimBA, SimBOINC, and EmBOINC are reviewed and description in detail.

Durrani and Shamsi [21] review the following VC problems: measurement of resource availability; determination of availability patterns; improved control of the resources using these patterns. Some naive and knowledge-based algorithms are described together with node grouping methods, security threats in VC, and safety measures.

In the survey [22] Khan et al. consider and compare popular modern middleware systems for DG: BOINC, XtremWeb, OurGrid, and HTCondor; they discuss the factors that

TABLE 2
Optimization Criteria and Methods used for Desktop Grid Scheduling

	Throughput (all results)	Throughput (valid results)	Makespan	Turnaround time	Fraction of incorrect results	Fraction of returned results	Overhead due to replication	Overhead due to deadlines	Overhead due to failures	Client load	Server load	Simulation & emulation	Grouping & clustering	Reputation & trust	Replication & voting
	Criteria											Methods			
Hwang et al, 2016 [93]	✓								✓			•			
Chernov and Nikitina, 2015 [32]					✓				✓						•
Mazalov et al., 2015 [73]	✓										✓				
Yasuda et al., 2015 [31]	✓				✓							•	•		
Alieksieiev et al., 2014 [57]			✓												
Canon et al., 2014 [91]			✓						✓						
Essafi et al., 2014 [58]			✓						✓					•	
Gil et al., 2014 [59]			✓			✓			✓				•		
Khan et al., 2014 [76]							✓						•		•
Mazalov et al., 2014 [74]											✓		•		
Miyakoshi et al., 2014 [30]					✓	✓						•		•	•
Rumiantsev, 2014 [60]			✓										•		•
Ujhelyi et al., 2014 [61]			✓									•	•		
Durrani et al., 2013 [55]	✓					✓							•		
Smaoui and Garbey, 2013 [62]			✓									•			
Klejnowski et al., 2012 [77]								✓						•	
Anderson, 2011 [47]								✓		✓	✓	•			
Bouguerra et al., 2011 [63]			✓						✓						
Canon et al., 2011a [64]			✓						✓						
Canon et al., 2011b [29]					✓		✓								
Watanabe et al., 2011 [28]	✓				✓							•	•	•	•
Al-Azzoni and Down, 2010 [71]				✓								•	•		
Celaya and Marchal, 2010 [65]			✓												
Lee et al., 2010 [66]			✓			✓						•			
Atlas et al., 2009 [43]	✓					✓						•			
Desell et al., 2009 [78]							✓								
Heien et al., 2009 [72]						✓									•
Anglano and Canonico, 2008 [44]				✓								•			
Estrada et al., 2008 [56]		✓					✓					•			
Byun et al., 2007 [67]			✓			✓									
Gao and Malewicz, 2007 [27]			✓		✓										
Kondo et al., 2007 [75]								✓		✓		•			
Sonnek et al., 2007 [37]		✓											•	•	•
Taufer et al., 2007a [35]		✓										•			
Taufer et al., 2007b [34]		✓										•		•	
Estrada et al., 2006 [33]		✓										•		•	
Fujimoto and Hagihara, 2006 [69]			✓												
Sonnek et al., 2006 [36]		✓											•	•	•
Choi et al., 2004 [70]			✓	✓											•
Kondo and Casanova, 2004 [68]			✓												
Maheswaran et al., 1999 [53]			✓									•	•		

influence the DG performance (from the client and the server sides) and review papers devoted to them.

3.4 Scheduling Research Papers

Here we present an overview of research papers related to scheduling in DGs. The optimization criteria and methods (see Section 3.2) used in the papers are summarized in Table 2.

Kondo et al. [75] compare, using a simulator, several scheduling policies for multi-project DG.

Availability of the nodes, if known or predictable, is often used to improve DG performance. Kondo and Casanova [68] propose a greedy-type scheduling algorithm and prove that it is optimal (with the minimal makespan) provided that availability intervals of DG nodes are known; they also test the algorithm on traces of a BOINC project.

Byun et al. [67] propose a Markov chain scheduler: a day consists of 48 half-hour intervals and each node can be idle, busy, or off during each interval, so that the availability state is a Markov chain. Three schemes (optimistic with no deadlines, pessimistic with tough deadlines, and a realistic one as a combination of the two) are considered. Efficiency of the method is checked experimentally and also proven analytically under the assumption that availability follows the hyper-exponential distribution.

Canon et al. [64] schedule tasks onto availability intervals of identical nodes, known in advance. The uncertainty is defined as the possible shift of an availability interval. Beside the analytic estimations including the worst case scenario, the approach is tested using simulations. In [91], the authors continue their analytic research of the impact of uncertainties on scheduling efficiency. They define the conditions on a scheduling algorithm to reach at least the same efficiency as predicted. The authors consider the nodes uniform rather than identical, propose a set of heuristics and thoroughly analyze them. The algorithms are tested using simulations on real BOINC projects traces.

Choi et al. [70] address the problem of performance decay due to scheduled and random shutdowns of DG nodes.

Availabilities of the nodes are monitored: a characteristic related to the reputation and a pulse check is used. The adaptive replication is applied. The authors give an analytical estimation of the efficiency under the assumption of the exponential distribution of availability periods and the Poisson arrival of new nodes joining the DG.

Casanova et al. [52] study the scheduling problem in a DG with unreliable (volatile) nodes that can be on, off, or temporarily unavailable. Bandwidth capacity of the network is taken into account. The problem is shown to be NP-complete even with full information on nodes' availability if the bandwidth is finite (but polynomial otherwise). The mean makespan is estimated under the Markovian assumption of the availability intervals, heuristic scheduling algorithms are discussed and tested using a simulator.

Replication is another wide area of study. Canon et al. [29] consider active saboteurs that are able to return the same wrong result on purpose. An additional assumption is the high cardinality of the set of possible answers so that a random coincidence is unlikely. The optimal replication is shown to improve both the replication overhead and the number of wrong results.

Condo et al. [94] consider a special important case when the number of tasks is less than the number of computers (which is typical for the final stage of a large project). Three strategies of task replication with the performance of the nodes taken into account are experimentally tested: idle resources need to be used to decrease the makespan.

Sonnek et al. [36], [37] define reliability as the probability to return the correct answer before the deadline. It is updated using the frequency of proven correct answers. The correctness of an answer is determined by voting in groups of nodes. The group size is variable and depends on the reliability of the group members. A few heuristics for group formation are tested using simulation.

Thresholds for reliability and availability are used by Estrada et al. [33] to increase the throughput of valid results. The method is tested on the EmBOINC emulator.

The paper by Taufer et al. [35] deals with molecular docking (Docking@home). Using SimBA, the authors show that adaptive selection of DGnodes can improve the throughput. Three policies are compared: "first come first served", fixed thresholds, and dynamical thresholds.

Watanabe et al. [28], [30] and also Yasuda et al. [31] propose a sabotage-proof scheduling method for BOINC-based DGs. The method is adaptive, voting-based, with the reliability of nodes (called credibility) taken into account. Node grouping according to their expected credibility is used in order to reduce the number of tasks not completed by the deadline. The failures reduce performance due to replication overhead, mostly. A similar idea is developed by Klejnowski et al. [77]: the waste of time due to failures to complete a task before the deadline is reduced by taking into account the nodes' reliability and trust increased by the successful past work.

Essafi et al. [58] develop a scheduling policy to minimize the difference between the actual total time to complete a project and the time in case all nodes were always available. The reputation of the nodes as the mean computing power is used to choose a node for a task.

Heien et al. [72] describe the PULL model of task scheduling and aims at the minimal violation of deadlines. Zero violation is proven for the algorithm if all the nodes are completely reliable. For the cases of unreliable nodes and random requests to the server, replication is used to improve the overall reliability.

Khan et al. [76] group nodes according to their capacity and the history of their work. The tasks scheduled to unreliable groups are replicated.

To schedule tasks in a heterogeneous computing system, Al-Azzoni and Down [71] group the tasks as well as the nodes according to their complexity/capacity and solve the linear optimization problem to find the maximal throughput with no task accumulation and enough computational power given to each task group. Additional restriction on the number of zero components in the solution makes the method more stable and practically useful. A similar approach is developed by Ujhelyi [61]: the nodes and the tasks are grouped by their dynamically evaluated performance and complexity; the approach is shown to reduce the total time and the "tail" phase (the time interval when there remain fewer tasks than nodes).

Maheswaran et al. [53] compare, using simulation, heuristic dynamical scheduling algorithms: five schedule tasks individually, three group task and schedule parcels. Two individual algorithms and one group one are proposed by the authors. Nodes' availability and heterogeneity are taken into account.

Grouping nodes according to their attributes is utilized by Durrani and Shamsi [55]: weighted sums of some properties are used to classify the resources. Gil et al. [59] clusterize the nodes by their availability (a fraction of time the node has been available) and reliability (probability to complete a task in time). Then the tasks are scheduled to the best available group and the best node within it. A heterogeneous cluster environment made of desktop computers is described by Jung et al. [88]. Some resources are available all time, other grant only part of their time. Most attention is paid to managing Linux and Windows resources.

Rumiantsev [60] uses task replication, grouping, and penalties for errors to reduce the makespan of a computational project in a DG. We studied [32] replication that minimizes the expected cost (consisting of the average time per a task and the penalties for wrong answers).

Bouguerra et al. [63] deal with unavailability by the optimal scheduling of checkpoints (so that uncompleted task can be continued on another node). The algorithm is shown to be minimal wasted time-optimal for any probability distribution of the nodes unavailability periods.

Fujimoto and Hagihara [69] propose an algorithm for scheduling n tasks onto m computers and prove the efficiency estimation analytically for the case of varying computational power, which is a DG feature. They also propose a new criterion, TPCC (total processor cycle consumption), which is directly related to the makespan.

Several papers address genetic algorithms applied to scheduling in DGs. Genetic algorithms solve optimization problems by simulating evolving populations of the variables being optimized. Those individuals who are better with respect to the utility function produce more descendants than the others. Mutations, i.e., changes of an individual, provide search in the neighborhood of a successful individual. Algorithms may differ in the mechanism of mutations, the type of

TABLE 3
Traces Information used in Papers on DG Scheduling

Paper	Key idea	Trace type	Trace data
Chernov and Nikitina, 2015 [32]	Replication minimizes the average cost including penalties for errors.	EDG	Workload characteristics
Mazalov et al., 2015 [73]	Task grouping to reduce the server load, game-theoretic approach	EDG, VC	Workload characteristics
Essafi et al., 2014 [58]	Makespan is minimized in presence of unavailability using availability reputation of nodes.	VC	Availability information, workload characteristics
Ujhelyi et al., 2014 [61]	Grouping the nodes by performance and the tasks by complexity.	EDG	Availability information
Canon et al., 2014 [91]	The conditions for a scheduling algorithm to lead to a lower-bound efficiency are defined.	VC	Availability information, workload characteristics
Canon et al., 2011a [64]	Tasks are scheduled over availability intervals of the nodes.	VC	Availability information, workload characteristics
Canon et al., 2011b [29]	Group result certification in the presence of cooperative saboteurs; replication reveals cheating.	VC	Availability information, workload characteristics
Desell et al., 2009 [78]	Verification of a limited subset of important (promising) results.	VC	Application model
Heien et al., 2009 [72]	Time before the pull recommended to nodes to respect deadlines, replication versus unreliability of the nodes.	VC	Availability information
Kondo et al., 2007 [94]	Full information, tail computing, the PUSH model. Task duplication is shown to be a good heuristics.	EDG	Availability information

Trace type: EDG = Enterprise DG; VC = volunteer computing.

reproduction, or the mechanism of concurrency between individuals.

In [56], Estrada et al. use genetic algorithms to produce the optimal scheduling policies. Sets of rules are individuals. Rules are described by a formal grammar and include inequalities, logical and arithmetical operations, and predefined variables, e.g., the node availability, productivity, etc. The output of the algorithm is the scheduling policy: a set of rules to choose a node for a task. A similar approach is used by Estrada and Taufer [20], where elitism and crossover operations are applied to individuals: the best ones pass to the new generation without mutations, and a pair of individuals join together to produce two descendants.

Qu et al. [95] and Wang et al. [54] propose and develop look-ahead genetic algorithms for task-to-nodes mapping that improve both makespan and reliability.

Smaoui and Garbey [62] address the problem of better scheduling of the tasks that use genetic algorithms in DGs. The apparent challenges of GA applications are the interdependence and the large size of generations: any fault may delay evaluation of a generation so that the next generation needs to wait. The approach ignores late individuals of a generation restoring its size by the best individuals (the elitism concept). Also, most promising individuals are evaluated on the best nodes available. In [78] a similar question is investigated by Desell et al. In particular, only individuals that are promising for improving the population quality are replicated and/or checked. In [57] a metaheuristic genetic algorithm is used for scheduling in DG.

In general, Table 2 illustrates the shift from early scheduling models with simple optimization criteria to more complex ones that consider multiple non-trivially interdependent optimization criteria and address intrinsic DG mechanisms influencing the scheduling. For instance, the throughput of valid results seems to become less popular since around 2008 [56], in favor of not only the closely related makespan parameter, but rather the mechanisms influencing the throughput: the overhead due to failures, the fractions of incorrect or never returned results, etc. Reasons for that could be problems with the exact definition of validity and inclusion of the verification into the scheduling scheme.

Table 3 presents the works that used traces of real BOINC projects in order to evaluate their scheduling algorithms. One can notice that Table 3 and Table 1 have only 6 papers in common. Therefore, there is an evident lack of analytically proven optimal scheduling algorithms with established practical efficiency.

3.5 Non-Regular Desktop Grids

As it was mentioned in Section 2.1, there are a number of non-regular DGs, like peer-to-peer or hierarchical ones, being developed by various research groups. Promising certain advantages, these non-regular DGs face new scheduling complexities. In particular, solving a set of interdependent tasks is a more complicated problem from the point of view of scheduling. For example, in a bottleneck case of a task B that depends recursively on almost one half of other tasks and the other half depends on B, all nodes need to wait for B to be solved (so it can be replicated many times with no loss of performance). Another case of linearly dependent tasks makes their parallel solving impossible. Usually, the structure of dependencies is apriori known as a directed acyclic graph. In some works, it is an arbitrary DAG, in others, it has a fixed shape. Peer-to-peer (P2P) grids are able to improve scheduling efficiency using additional horizontal links; they are especially useful for solving dependent tasks. Also using multi-

level grids can improve scheduling due to more efficient local control.

In this section we exemplify how the horizontal P2P links and multiple scheduling levels are used by researchers to improve the scheduling efficiency. Then we cite several contributions that focus on scheduling interdependent tasks in DGs. We do not claim to survey all works in this area but try to present the most important methods and approaches.

The Volpex (Parallel Execution on Volatile nodes) project [96], [97] is a Message Passing Interface (MPI) implementation for DGs. It joins the computing nodes to a virtual cluster suitable for applications with moderate communication requirements. Being based on MPI, this approach can address both P2P communication and possible dependence between the tasks. The implementation takes into account challenges of DGs: heterogeneity and unreliability.

In [98], Kwan and Muppala analyze a P2P DG. There is a set of super-seeds that have a permanent high-speed connection with each other. The super-seeds store information on network segments. They also assign tasks to volunteer workers. A simple mechanism is used: the first task in a queue is assigned to the most powerful among vacant workers. The authors study the properties of the fixed/adaptive threshold gossiping mechanism.

In the same fashion, Rius et al. [99] consider task scheduling in two-level peer-to-peer computational DGs, where super peers observe subsets of computing nodes. Prior to final task distribution, the tasks are grouped basing on local neighborhood information only. Then the intermediate scheduling decisions are exchanged between super peers to form the final schedule.

Murata et al. [100] propose a distributed and cooperative load balancing mechanism. It implements horizontal scheduling; each computing node can exchange its tasks with the others so as to minimize the execution time by dynamic load balancing among the computing nodes.

In [65], Celaya and Marchal consider the problem of fair resource sharing among multiple applications in a distributed computing platform. The platform itself is a P2P DG with a tree-like structure: the leaves do computing, while other nodes serve as communicators. The information about the availability of the nodes goes towards the root, while the information about the tasks goes in the opposite direction. The "earliest deadline first" local scheduling policy is being used. A global scheduling policy is the following. When a branch node receives a request for the allocation of a new application, it calculates the minimum application stretch (slowdown) in its own sub-tree based on availability summaries. The application is either scheduled in the subtree or going up the tree to achieve a better stretch. Once a routing node finds a suitable stretch, the tasks are distributed among the children nodes. This implements a trade-off between performance and load-balancing: small applications remain local, while large ones go up the tree until they induce an acceptable slowdown.

Hwang et al. [93] compare three resource allocation policies for multi-user and multi-application workloads in a heterogeneous computing system for solving loosely coupled tasks, trading between efficiency and fairness.

The paper of Chmaj et al. [90] describes a peer-to-peer DG in which all the computing nodes want to receive the results of all the tasks computation. The authors use a BitTorrent-like scheme for results sharing, where one of the nodes acts as a tracker as it provides the information on results location and distributes the tasks. The objective is formulated as an integer linear programming optimization problem where binary variables indicate whether to assign a task to a node or whether to transfer a result to a node. The authors assume that all the transfer costs are known, all the tasks should be assigned, all the nodes should have all the results, each node should solve at least one task and not more than a specified limit. The problem is NP-hard. The authors provide two heuristics: a greedy and a genetic one.

A related yet different approach is applied by Hussin et al. [102]: the target system consists of a number of resource sites that are loosely connected by a communication network. Each site has a set of heterogeneous processing nodes that are fully interconnected and composed of a different number of cores with homogeneous processing speed. There is a centralized scheduler to handle tasks from system users and map them onto processing nodes. Users' tasks are considered independent (i.e., no inter-job communication or dependencies take place), and their time to arrive at a scheduler is not known in advance. Two parameters used to evaluate each task are the computational complexity and the type of task. There are two different types of tasks: regular and special (with special requirements for reliability, network speed, etc.). The task execution time varies according to the performance of the resource where the task is being processed. The authors assume that the task profile is available. Based on the known performance and estimated availability of the nodes, the authors propose a special scheduling mechanism, which keeps two separate queues with a dynamic tasks redistribution.

Farkas and Kacsuk [104] describe the hierarchical DG MTA SZTAKI where DGs serve as nodes and perform their own scheduling. The authors consider five different scheduling algorithms varying from the simple accounting of a number of clients to more complex accounting of the active client timeout. The authors use the Hierarchical DG Simulator to evaluate some metrics like makespan and deadline violations among three scenarios: non-hierarchical DG, two-level hierarchical DG with one or two clients.

Celaya and Arronategui [101] were motivated by hierarchical DGs to develop a distributed dynamic scheduling algorithm suitable for very big computing networks. The computing network itself is an hierarchy where each node is a computing and routing node at the same time. The scheduling algorithm includes information on the nodes availability propagation mechanism. The approach is checked on three different policies using simulation.

One of the strongest assumptions of the regular DGs is the tasks independence, i.e., DGs are most suitable for BoT applications. Processing interdependent tasks is possible, but this severely constrains the scheduler: switch-off of unreliable nodes can spoil the performance much more compared to the BoT case; a wrong result needs to be recalculated with all its dependency chain; the "bottlenecks", i.e., the tasks that are waited for by many other tasks, can drastically decrease performance, so that careful scheduling in order to avoid bottlenecks is necessary. Dependence means that a task can only be executed when all its ancestors already have been.

TABLE 4
Papers on Scheduling in Non-Regular DGs

Paper	Key idea	P2P DG	Hierarchical DG	Interdependent tasks
Celaya and Arronategui, 2013 [101]	Two-level scheduling, tree-shaped DG, each node serves as a computer, a server, and a router.		√	
Hussin et al., 2013 [102]	Two queues for the tasks of two kinds (normal and with special demands).	√	√	
Rius et al., 2013 [99]	Two-level superpeer scheduling.	√	√	
Chmaj et al., 2012 [90]	Torrent-like P2P task scheduling and answer propagation. Binary linear optimization. Greedy and genetic heuristics.	√		
Cordasco et al., 2012 [103]	Dependent tasks with known DAG, maximizing the average number of tasks ready for solving.			√
Farkas and Kacsuk, 2012 [104]	Hierarchical DGs (a client is a DG itself): performance evaluation of five algorithms.		√	
Celaya and Marchal, 2010 [65]	Tree-shaped network, multiple batches, flow of tasks/availability is downwards/upwards.	√	√	
Kwan and Muppala, 2010 [98]	Optimizing inter-segment communications.	√	√	
Lee et al., 2010 [66]	Greedy optimization of the delay time by task redistribution.	√		√
Qu et al., 2010 [95]	Evolution on task-node mapping according to resource priorities.			√
Murata et al., 2008 [100]	Load balancing, node grouping, decentralized task redistribution.	√		
Gao and Malewicz, 2004 [105]	Rectangular graph of task dependencies, unreliable nodes with a reliable server, optimal validation.			√

Qu et al. [95] consider an arbitrary DAG. The authors propose an algorithm to optimize both time and reliability for a workflow application. The algorithm uses a special evolution and evaluation mechanism: the evolution operators evolve the task-resource mapping for a scheduling solution, while the solution's task order is determined in the evaluation step using the max-min strategy proposed by the authors. The authors do not describe the computational system and tasks characteristics (including DAG) but compare their algorithm with several others.

In paper [103] Cordasco et al. propose a new cost function called the *AREA* to define the number of tasks ready for execution at a given time moment. The authors try to find out that schedules having higher *AREAs* would tend to have computational benefits - notably, smaller makespans - when executing DAGs. The experimental results do not establish such a correlation with any statistical certainty, but they do suggest that the correlation may exist - at least for certain families of DAGs and certain patterns of the worker availability. In the paper it is shown that the *AREA* metric for DAG-scheduling enjoys several algorithmically significant properties. The authors developed conceptual and algorithmic tools for crafting efficient area-maximizing schedules for a variety of DAG-types, including monotonic tree-DAGs, expansive-reductive DAGs, and compositions of either bipartite cliques or cycles. The authors extend the model in [106] for multithreading applications, and in [107] they present an efficient scheduling heuristic for maximizing the *AREA*.

Gao et al. [27] consider a DG of unreliable nodes with the reliable server solving dependent tasks with a special DAG of dependencies: it is a square lattice where each task depends on its right and top neighbor. The problem is to choose which tasks to check so as to maximize the average number of the tasks completed before the deadline. The complexity of the problem is estimated, and the algorithms for special cases of high and low reliability are proposed.

The paper by Lee et al. [66] is devoted to the robustness (the stability with respect to changes of the DG) of the scheduling. The tasks are dependent, their DAG of dependencies is known. Two greedy scheduling policies are proposed and tested numerically: they re-schedule tasks to different nodes in order to maximize either the total allowable delay time or the minimum delay time.

The non-regular DG papers are summarized in Table 4.

Acknowledgments

This work is supported by the Russian Foundation for Basic Research [project 16-07-00622].

References

[1] M. Litzkow, "Remote Unix—turning idle workstations into cycle servers," in *Proc. Usenix Summer Conf.*, 1987, pp. 381–384.
[2] L. F. Sarmenta and S. Hirano, "Bayanihan: Building and studying web-based volunteer computing systems using Java," *Future Generation Comput. Syst.*, vol. 15, no. 5, pp. 675–686, 1999.

[3] I. Foster, C. Kesselman, and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," *The Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200–222, 2001.

[4] Great internet mersenne prime search GIMPS, [Online]. Available: https://www.mersenne.org, Accessed on: Apr. 19, 2018.

[5] distributed.net, [Online]. Available: www.distributed.net, Accessed on: Apr. 19, 2018.

[6] W. T. Sullivan, D. Werthimer, S. Bowyer, J. Cobb, D. Gedye, and D. Anderson, "A new major SETI project based on Project SERENDIP data and 100,000 personal computers," in *Proc. Int. Astron. Union Colloq.*, 1997, pp. 729–734.

[7] Folding@home, [Online]. Available: folding.stanford.edu, Accessed on: Apr. 19, 2018.

[8] D. P. Anderson, "BOINC: A system for public-resource computing and storage," in *Proc. 5th IEEE/ACM Int. Workshop Grid Comput.*, 2004, pp. 4–10.

[9] G. Fedak, C. Germain, V. Neri, and F. Cappello, "XtremWeb: A generic global computing system," in *Proc. 1st IEEE/ACM Int. Symp. Cluster Comput. Grid*, 2001, pp. 582–587.

[10] M. Merz, K. Muller, and W. Lamersdorf, "Service trading and mediation in distributed computing systems," in *Proc. 14th Int. Conf. Distrib. Comput. Syst.*, 1994, pp. 450–457.

[11] M. Filamofitsky, "The system X-Com for metacomputing support: Architecture and technology," *Vychislitel'nye Metody i Programmirovanie*, vol. 5, no. 2, pp. 1–9, 2004.

[12] A. Chien, B. Calder, S. Elbert, and K. Bhatia, "Entropia: Architecture and performance of an enterprise desktop grid system," *J. Parallel Distrib. Comput.*, vol. 63, no. 5, pp. 597–610, 2003.

[13] B. O. Christiansen, P. Cappello, M. F. Ionescu, M. O. Neary, K. E.Schauser, and D. Wu, "Javelin: Internet-based parallel computing using Java," *Concurrency: Practice Experience*, vol. 9, no. 11, pp. 1139–1160, 1997.

[14] M. J. Litzkow, M. Livny, and M. W. Mutka, "Condor — A hunter of idle workstations," in *Proc. 8th Int. Conf. Distrib. Comput. Syst.*, 1988, pp. 104–111.

[15] Univa grid engine, [Online]. Available: www.univa.com/products, Accessed on: Apr. 19, 2018.

[16] A. Luther, R. Buyya, R. Ranjan, and S. Venugopal, "Alchemi: A. NET-based enterprise grid computing system," in *Proc. Conf. Internet Comp.*, 2005, pp. 269–278.

[17] BOINCstats, [Online]. Available: https://boincstats.com, Accessed on: Apr. 19, 2018.

[18] S. Choi, H. S. Kim, E. J. Byun, M. S. Baik, S. S. Kim, C. Y. Park, and C. S. Hwang, "Characterizing and classifying desktop grid," in *Proc. 7th IEEE Int. Symp. Cluster Comput. Grid*, 2007, pp. 743–748.

[19] F. Xhafa and A. Abraham, "Computational models and heuristic methods for grid scheduling problems," *Future Generation Comput. Syst.*, vol. 26, no. 4, pp. 608–621, Apr. 2010.

[20] T. Estrada and M. Taufer, "Challenges in designing scheduling policies in volunteer computing," in *Desktop Grid Computing*, C. Cérin and G. Fedak, Eds. Boca Raton, FL, USA: CRC Press, 2012, pp. 167–190.

[21] N. Durrani and J. Shamsi, "Volunteer computing: Requirements, challenges, and solutions," *J. Netw. Comput. Appl.*, vol. 39, pp. 369–380, Mar. 2014.

[22] M. Khan, T. Mahmood, and S. Hyder, "Scheduling in desktop grid systems: Theoretical evaluation of policies and frameworks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 1, pp. 119–127, 2017.

[23] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. P. Anderson, "Cost-benefit analysis of cloud computing versus desktop grids," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2009, pp. 1–12.

[24] A. Afanasiev, I. Bychkov, M. Manzyuk, M. Posypkin, A. Semenov, and O. Zaikin, "Technology for integrating idle computing cluster resources into volunteer computing projects," in *Proc. 5th Int. Workshop Comput. Sci. Eng.*, 2015, pp. 109–114.

[25] J. Kovács, A. C. Marosi, Á. Visegrádi, Z. Farkas, P. Kacsuk, and R. Lovas, "Boosting gLite with cloud augmented volunteer computing," *Future Generation Comput. Syst.*, vol. 43, pp. 12–23, 2015.

[26] D. Montes, J. A. Añel, T. F. Pena, P. Uhe, and D. C. Wallom, "Enabling BOINC in infrastructure as a service cloud system," *Geoscientific Model Develop.*, vol. 10, no. 2, 2017, Art. no. 811.

[27] L. Gao and G. Malewicz, "Toward maximizing the quality of results of dependent tasks computed unreliably," *Theory Comput. Syst.*, vol. 41, no. 4, pp. 731–752, 2007.

[28] K. Watanabe, M. Fukushi, and M. Kameyama, "Adaptive group-based job scheduling for high performance and reliable volunteer computing," *J. Inf. Process.*, vol. 19, pp. 39–51, 2011.

[29] L.-C. Canon, E. Jeannot, and J. Weissman, "A scheduling and certification algorithm for defeating collusion in desktop grids," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 343–352.

[30] Y. Miyakoshi, S. Yasuda, K. Watanabe, M. Fukushi, and Y. Nogami, "Dynamic job scheduling method based on expected probability of completion of voting in volunteer computing," *IEICE Trans. Inf. Syst.*, vol. 98, no. 12, pp. 2132–2140, 2015.

[31] S. Yasuda, Y. Nogami, and M. Fukushi, "A dynamic job scheduling method for reliable and high-performance volunteer computing," in *Proc. 2nd Int. Conf. Inf. . Security*, 2015, pp. 1–4.

[32] I. Chernov and N. Nikitina, "Virtual screening in a desktop grid: Replication and the optimal quorum," in *Proc. Int. Conf. Parallel Comput. Technol.*, 2015, pp. 258–267.

[33] T. Estrada, D. Flores, M. Taufer, P. Teller, A. Kerstens, and D. Anderson, "The effectiveness of threshold-based scheduling policies in BOINC projects," in *Proc. 2nd IEEE Int. Conf. e-Sci. Grid Comput.*, 2006, pp. 88–88.

[34] M. Taufer, A. Kerstens, T. Estrada, D. Flores, and P. Teller, "SimBA: A discrete event simulator for performance prediction of volunteer computing projects," in *Proc. 21st Int. Workshop Principles Adv. Distrib. Simul.*, 2007, pp. 189–197.

[35] M. Taufer, A. Kerstens, T. Estrada, D. Flores, R. Zamudio, P. Teller, R. Armen, and C. L. Brooks III, "Moving volunteer computing towards knowledge-constructed, dynamically-adaptive modeling and scheduling," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2007, pp. 1–8.

[36] J. Sonnek, M. Nathan, A. Chandra, and J. Weissman, "Reputation-based scheduling on unreliable distributed infrastructures," in *Proc. 26th IEEE Int. Conf. Distrib. Comput. Syst.*, 2006, pp. 30–30.

[37] J. Sonnek, A. Chandra, and J. Weissman, "Adaptive reputation-based scheduling on unreliable distributed infrastructures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 11, pp. 1551–1564, Nov. 2007.

[38] [Online]. Available: https://boinc.berkeley.edu/trac/wiki/, Accessed on: Apr. 19, 2018.

[39] A. L. Bazinet and M. P. Cummings, "Computing the tree of life: Leveraging the power of desktop and service grids," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops Phd Forum*, May 2011, pp. 1896–1902.

[40] P. Hellinckx, S. Verboven, F. Arickx, and J. Broeckhove, "Predicting parameter sweep jobs: From simulation to grid implementation," in *Proc. Int. Conf. Complex Intell. Softw. Intensive Syst.*, Mar. 2009, pp. 402–408.

[41] P. Hellinckx, S. Verboven, F. Arickx, and J. Broeckhove, "Runtime prediction in desktop grid scheduling," *Parallel Program. Appl. Grid, P2P Netw.-Based Syst.*, vol. 1, pp. 204–231, Jan. 2009.

[42] Y. Kolokoltsev, E. Ivashko, and C. Gershenson, "Improving "tail" computations in a BOINC-based desktop grid," *Open Eng.*, vol. 7, no. 1, pp. 371–378, 2017.

[43] J. Atlas, T. Estrada, K. Decker, and M. Taufer, "Balancing scientist needs and volunteer preferences in volunteer computing using constraint optimization," in *Proc. 9th Int. Conf. Comput. Sci.*, pp. 143–152, 2009.

[44] C. Anglano and M. Canonico, "Scheduling algorithms for multiple bag-of-task applications on desktop grids: A knowledge-free approach," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2008, pp. 1–8.

[45] C. A. and. M. Canonico, "Fault-tolerant scheduling for bag-of-tasks grid applications," in *Proc. Eur. Grid Conf.*, 2005, pp. 630–639.

[46] D. da Silva, W. Cirne, and F. Brasileiro, "Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids," in *Proc. Eur. Conf. Parallel Process.*, 2003, pp. 169–180.

[47] D. Anderson, "Emulating volunteer computing scheduling policies," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops Phd Forum*, 2011, pp. 1839–1846.

[48] O. Beaumont, L. Bobelin, H. Casanova, P.-N. Clauss, B. Donassolo, L. Eyraud-Dubois, S. Genaud, S. Hunold, A. Legrand, M. Quinson, et al., "Towards scalable, accurate, and usable simulations of distributed applications and systems," Institut National de Recherche en Informatique et en Automatique, France, Tech. Rep. RR-7761, 2011.

[49] T. Estrada, M. Taufer, and D. Anderson, "Performance prediction and analysis of BOINC projects: An empirical study with EmBOINC," *J. Grid Comput.*, vol. 7, no. 4, pp. 537–554, Aug. 2009.

[50] T. Estrada, M. Taufer, K. Reed, and D. Anderson, "EmBOINC: An emulator for performance analysis of BOINC projects," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2009, pp. 1–8.

[51] S. Alonso-Monsalve, F. Garca-Carballeira, and A. Caldern, "ComBos: A complete simulator of volunteer Computing and Desktop Grids," *Simul. Modelling Practice Theory*, vol. 77, pp. 197–211, 2017.

[52] H. Casanova, F. Dufossé, Y. Robert, and F. Vivien, "Scheduling parallel iterative applications on volatile resources," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, May 2011, pp. 1012–1023.

[53] M. Maheswaran, S. Ali, H. Siegel, D. Hensgen, and R. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *J. Parallel Distrib. Comput.*, vol. 59, no. 2, pp. 107–131, 1999.

[54] X. Wang, C. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Comput. Syst.*, vol. 27, no. 8, pp. 1124–1134, Oct. 2011.

[55] N. K. Nouman, M. Durrani, J. A. Shamsi, and S. Fatima, "Towards efficient resource grouping in heterogeneity-aware volunteer computing," *Sindh Univ. Res. J.*, vol. 45, no. 3, pp. 1–11, 2013.

[56] T. Estrada, O. Fuentes, and M. Taufer, "A distributed evolutionary method to design scheduling policies for volunteer computing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 3, pp. 40–49, 2008.

[57] M. Alieksieiev, O. Maiboroda, A. Onysko, and O. Alekseev, "Repeating tasks scheduling in desktop grid systems," in *Proc. 24th Int. Crimean Conf. Microw. Telecommun. Technol.*, 2014, pp. 342–343.

[58] A. Essafi, D. Trystram, and Z. Zaidi, "An efficient algorithm for scheduling jobs in volunteer computing platforms," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp. Workshops*, May 2014, pp. 68–76.

[59] J.-M. Gil, S. Kim, and J. Lee, "Task scheduling scheme based on resource clustering in desktop grids," *Int. J. Commun. Syst.*, vol. 27, no. 6, pp. 918–930, Jun. 2014.

[60] A. Rumiantsev, "Optimizing the execution time of a desktop grid project," *Program Syst.: Theory Appl.*, vol. 5, pp. 175–182, 2014.

[61] M. Ujhelyi, P. Lacko, and A. Paulovic, "Task scheduling in distributed volunteer computing systems," in *Proc. IEEE 12th Int. Symp. Intell..*, 2014, pp. 111–114.

[62] M. Smaoui and M. Garbey, "Improving volunteer computing scheduling for evolutionary algorithms," *Future Generation Comput. Syst.*, vol. 29, no. 1, pp. 1–14, Jan. 2013.

[63] M. S. Bouguerra, D. Kondo, and D. Trystram, "On the scheduling of checkpoints in desktop grids," in *Proc. 11th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2011, pp. 305–313.

[64] L.-C. Canon, A. Essafi, G. Mounié, and D. Trystram, "A bi-objective scheduling algorithm for desktop grids with uncertain resource availabilities," in *Proc. 17th Int. Conf. Parallel Process.*, 2011, pp. 238–249.

[65] J. Celaya and L. Marchal, "A fair decentralized scheduler for bag-of-tasks applications on desktop grids," in *Proc. 10th IEEE/ACM Int. Conf. Cluster Cloud Grid Comput.*, 2010, pp. 538–541.

[66] Y. Lee, A. Zomaya, and H. Siegel, "Robust task scheduling for volunteer computing systems," *J. Supercomput.*, vol. 53, no. 1, pp. 163–181, Jul. 2010.

[67] E. Byun, S. Choi, M. Baik, J. Gil, C. Park, and C. Hwang, "MJSA: Markov job scheduler based on availability in desktop grid computing environment," *Future Generation Comput. Syst.*, vol. 23, pp. 616–622, 2007.

[68] D. Kondo and H. Casanova, "Computing the optimal makespan for jobs with identical and independent tasks scheduled on volatile hosts," Dept. Comput. Sci. Eng., Univ. California, San Diego, CA, Tech. Rep. CS2004–0796, 2004.

[69] N. Fujimoto and K. Hagihara, "A 2-approximation algorithm for scheduling independent tasks onto a uniform parallel machine and its extension to a computational grid," in *Proc. IEEE Int. Conf. Cluster Comput.*, 2006, pp. 1–7.

[70] S. Choi, M. Baik, C. Hwang, J. Gil, and H. Yu, "Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment," in *Proc. 3rd IEEE Int. Symp. Netw. Comput. Appl.*, 2004, pp. 366–371.

[71] I. Al-Azzoni and D. Down, "Dynamic scheduling for heterogeneous desktop grids," *J. Parallel Distrib. Comput.*, vol. 70, no. 12, pp. 1231–1240, Dec. 2010.

[72] E. Heien, D. Anderson, and K. Hagihara, "Computing low latency batches with unreliable workers in volunteer computing environments," *J. Grid Comput.*, vol. 7, no. 4, pp. 501–518, Dec. 2009.

[73] V. Mazalov, N. Nikitina, and E. Ivashko, "Task scheduling in a desktop grid to minimize the server load," in *Proc. Int. Conf. Parallel Comput. Technol.*, 2015, pp. 273–278.

[74] V. Mazalov, N. Nikitina, and E. Ivashko, "Hierarchical two-level game model for tasks scheduling in a desktop grid," in *Proc. 6th Int. Congress Ultra Modern Telecommun. Control Syst. Workshops*, 2014, pp. 541–545.

[75] D. Kondo, D. Anderson, and J. M. VII, "Performance evaluation of scheduling policies for volunteer computing," in *Proc. 3rd IEEE Int. Conf. e-Sci. Grid Comput.*, 2007, pp. 221–227.

[76] M. Khan, S. Hyder, G. Ahmed, S. Begum, and M. Aamir, "A group based replication mechanism to reduce the wastage of processing cycles in volunteer computing," *Wireless Pers. Commun.*, vol. 76, no. 3, pp. 591–601, Jun. 2014.

[77] L. Klejnowski, Y. Bernard, C. Muller-Schloer, and J. Hahner, "Using trust to reduce wasteful computation in open desktop grid systems," in *Proc. 10th Annu. Int. Conf. Privacy Security Trust*, 2012, pp. 250–255.

[78] T. Desell, M. Magdon-Ismail, B. Szymanski, C. Varela, H. Newberg, and N. Cole, "Robust asynchronous optimization for volunteer computing grids," in *Proc. 5th IEEE Int. Conf. e-Sci.*, Dec. 2009, pp. 263–270.

[79] 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 *Proc. 14th IEEE Int. Conf. Parallel Distrib. Syst.*, 2008, pp. 171–178.

[80] C. Li and L. Li, "Utility-based scheduling for grid computing under constraints of energy budget and deadline," *Comput. Standards Interfaces*, vol. 31, pp. 1131–1142, 2009.

[81] L. Ponciano and F. Brasileiro, "On the impact of energy-saving strategies in opportunistic grids," in *Proc. 11th IEEE/ACM Int. Conf. Grid Comput.*, 2010, pp. 282–289.

[82] S. Nesmachnow, B. Dorronsoro, J. Pecero, and P. Bouvry, "Energy-aware scheduling on multicore heterogeneous grid computing systems," *J. Grid Comput.*, vol. 11, pp. 653–680, 2013.

[83] A. Tchernykh, J. Pecero, A. Barrondo, and E. Schaeffer, "Adaptive energy efficient scheduling in peer-to-peer desktop grids," *Future Generation Comput. Syst.*, vol. 36, pp. 209–220, 2014.

[84] S. Kianpisheh, M. Kargahi, and N. M. Charkari, "Resource availability prediction in distributed systems: An approach for modeling non-stationary transition probabilities," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 8, pp. 2357–2372, 2018.

[85] Y. Shang and L. Shang, "Trust model for reliable node allocation based on daily computer usage behavior," *Concurrency Comput.*, vol. 30, no. 6, 2018, Art. no. e4346.

[86] M. Khan and T. Mahmood, "Using predictive analytics for predicting host availability in desktop grids," *Int. J. Grid Distrib. Comput.*, vol. 10, no. 1, pp. 245–254, 2017.

[87] S. Rubab, M. Hassan, A. Mahmood, and S. Shah, "Proactive job scheduling and migration using artificial neural networks for volunteer grid," in *Proc. 1st EAI Int. Conf. Comput. Sci. Eng.*, 2017.

[88] D. Jung, H. Jeong, J. Kim, D. Lee, and M. Kim, "The resource running time manager for integrated environment," *Cluster Comput.*, pp. 1–10, 2017, https://doi.org/10.1007/s10586-017-1520-1

[89] E. Ivashko, "Mathematical model of a "tail" computation in a desktop grid," in *Proc. CEUR Workshop*, 2017, pp. 54–59.

[90] 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," *Int. J. Appl. Math. Comput. Sci.*, vol. 22, no. 3, pp. 733–748, 2012.

[91] L.-C. Canon, A. Essafi, and D. Trystram, "A proactive approach for coping with uncertain resource availabilities on desktop grids," in *Proc. 21st Int. Conf. High Perform. Comput.*, 2014, pp. 1–9.

[92] N. Nikitina, E. Ivashko, and A. Tchernykh, "Congestion game scheduling for virtual drug screening optimization," *J. Comput.-Aided Mol. Des.*, vol. 32, no. 2, pp. 363–374, 2018.

[93] E. Hwang, S. Kim, T. Kyung Yoo, J.-S. Kim, S. Hwang, and Y.-R. Choi, "Resource allocation policies for loosely coupled applications in heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2349–2362, Aug. 2016.

[94] D. Kondo, A. Chien, and H. Casanova, "Scheduling task parallel applications for rapid turnaround on enterprise desktop grids," *J. Grid Comput.*, vol. 5, no. 4, pp. 379–405, Oct. 2007.
[95] B. Qu, Y. Lei, and Y. Zhao, "A new genetic algorithm based scheduling for volunteer computing," in *Proc. Int. Conf. Comput. Commun. Technol. Agriculture Eng.*, 2010, pp. 228–231.
[96] M. T. Rahman, H. Nguyen, J. Subhlok, and G. Pandurangan, "Checkpointing to minimize completion time for inter-dependent parallel processes on volunteer grids," in *Proc. 16th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, May 2016, pp. 331–335.
[97] T. LaBlanc, "Design and evaluation of a communication library for volunteer computing environments," Ph.D. dissertation, University of Houston, Houston, TX, 2009.
[98] S. Kwan and J. Muppala, "Bag-of-tasks applications scheduling on volunteer desktop grids with adaptive information dissemination," in *Proc. IEEE 35th Conf. Local Comput. Netw.*, 2010, pp. 544–551.
[99] J. Rius, F. Cores, and F. Solsona, "Cooperative scheduling mechanism for large-scale peer-to-peer computing systems," *J. Netw. Comput. Appl.*, vol. 36, no. 6, pp. 1620–1631, 2013.
[100] Y. Murata, T. Inaba, H. Takizawa, and H. Kobayashi, "Implementation and evaluation of a distributed and cooperative load-balancing mechanism for dependable volunteer computing," in *Proc. IEEE Int. Conf. Depend. Syst. Netw. FTCS DCC*, 2008, pp. 316–325.
[101] J. Celaya and U. Arronategui, "A task routing approach to large-scale scheduling," *Future Generation Comput. Syst.*, vol. 29, no. 5, pp. 1097–1111, 2013.
[102] M. Hussin, A. Abdullah, and S. Subramaniam, "Adaptive resource allocation for reliable performance in heterogeneous distributed systems," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, 2013, pp. 51–58.
[103] G. Cordasco, R. De Chiara, and A. Rosenberg, "On scheduling DAGs for volatile computing platforms: Area-maximizing schedules," *J. Parallel Distrib. Comput.*, vol. 72, no. 10, pp. 1347–1360, Oct. 2012.
[104] Z. Farkas and P. Kacsuk, "Evaluation of hierarchical desktop grid scheduling algorithms," *Future Generation Comput. Syst.*, vol. 28, no. 6, pp. 871–880, Jun. 2012.
[105] L. Gao and G. Malewicz, "Internet computing of tasks with dependencies using unreliable workers," in *Principles of Distributed Systems*. Berlin, Germany: Springer, 2004, pp. 443–458.
[106] G. Cordasco and A. L. Rosenberg, "On scheduling series-parallel DAGs to maximize AREA," *Int. J. Found. Comput. Sci.*, vol. 25, no. 05, pp. 597–621, 2014.
[107] G. Cordasco, R. De Chiara, and A. L. Rosenberg, "An AREA-oriented heuristic for scheduling DAGs on volatile computing platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 8, pp. 2164–2177, Aug. 2015.

Evgeny Ivashko received the PhD degree in math and physics, in 2010 and joined the researchers of the Institute of Applied Mathematical Research of Karelian Research Centre, Russian Academy of Sciences. He is an expert in the field of BOINC-based high-performance distributed computing. In 2011 he became a head of the new High-Performance Computing Center of KRC. Also he is an expert of the Skolkovo Fund and the head of one of the leading groups in Russia in the BOINC-based distributed computing. He is the author of papers in game theory and high-performance and distributed computing. The main topics of interest are the game theory, high-performance computing, distributed computing, artificial information systems.

Ilya Chernov received the PhD degree in mathematics and physics from Saint-Petersburg State University, Russia, in 2004. He has been working in the Institute of Applied Mathematical Research of KRC as a senior researcher and is an expert in the mathematical and numerical modeling of physical phenomena, including formation and decomposition of metal hydrides and large-scale sea circulation. Other research interests include numerical mathematics, high-performance computing, parabolic boundary-value problems of mathematical physics.

Natalia Nikitina received the PhD degree in technics from Petrozavodsk State University, Russia, in 2014. She is a researcher with the Institute of Applied Mathematical Research of KRC. Her main research interests are high-performance and distributed computing, parallel job scheduling, workload characterization and modeling, desktop grids, BOINC and applications of game theory.