Congestion Game Scheduling Implementation for High-Throughput Virtual Drug Screening Using BOINC-Based Desktop Grid

Natalia Nikitina[1(✉)], Evgeny Ivashko[1], and Andrei Tchernykh[2]

[1] Institute of Applied Mathematical Research, Karelian Research Center, Russian Academy of Sciences, Petrozavodsk, Russia
{nikitina, ivashko}@krc.karelia.ru
[2] Computer Science Department, CICESE Research Center, Ensenada, Baja California, Mexico
chernykh@cicese.mx

Abstract. Virtual drug screening is one of the most common applications of high-throughput computing. As virtual screening is time consuming, a problem of obtaining a diverse set of hits in a short time is very important. We propose a mathematical model based on game theory. Task scheduling for virtual drug screening in high-performance computational systems is considered as a congestion game between computing nodes to find the equilibrium solutions for best balancing between the number of interim hits and their chemical diversity. We present the developed scheduling algorithm implementation for Desktop Grid and Enterprise Desktop Grid, and perform comprehensive computational experiments to evaluate its performance. We compare the algorithm with two known heuristics used in practice and observe that game-based scheduling outperforms them by the hits discovery rate and chemical diversity at earlier steps.

Keywords: Drug discovery · Virtual drug screening · High-performance computing · High-throughput computing · Desktop grid · BOINC · Scheduling · Game theory · Congestion game

1 Introduction

Drug development is a time-consuming process. It takes up to 10–15 years to develop a new market-available drug [1]. One of the first stages of this process is an identification of a set of chemical compounds called *hits* with predicted desired biochemical activity. Hits are identified among a set of ligands, low-molecular compounds able to form a biochemical complex with a protein molecule responsible for disease progression, called a target.

With computer development of highly accurate specific disease models to validate targets and bind ligands to targets, virtual screening [2] (VS) has emerged to aid this stage of research. In the course of VS, one performs computer modeling of the interaction of the candidate ligands with the target and scores the resulting molecular complexes. The ligands with high scores become hits.

V. Malyshkin (Ed.): PaCT 2017, LNCS 10421, pp. 480–491, 2017.
DOI: 10.1007/978-3-319-62932-2_46

The chemical space of all small organic molecules, each of which might become a drug for the studied disease, is estimated to have the order of 10^{60} [3]. The libraries that are used in VS setups typically have size from hundreds of thousands to tens of millions of molecules [4–7]. The largest database GDB-17 contains more than 166 billion molecules [8]. Performing VS over such large databases essentially requires high-performance computational facilities. The exhaustive structure-based VS could take over five months at one of the world's fastest supercomputers Tianhe-2 [9].

With proper VS organization, the interim results can significantly contribute to the research progress boost. Considering the long time interval to screen the whole ligand library, it is important to obtain a diverse set of hits as soon as possible, so that they could proceed to the next stage of research in laboratory without finding the rest.

In this contribution, we propose a method for reducing the explored ligands space on the fly when performing structure-based VS to obtain diverse and useful results in a short time. The method is based on mathematical model of task scheduling. We develop an algorithm which promises the best balancing between the number of interim hits and their diversity. The algorithm does not depend on the docking algorithm implementation, the quality of protein and ligands models used for VS.

The model considers the heterogeneous environment with uncertainty of processing time and workload, and limited knowledge about the input dataset structure. The algorithm can be used to develop software based on a high-performance computing cluster, a Grid system or a Cloud. In this paper, we concentrate on the algorithm implementation for Desktop Grid and Enterprise Desktop Grid.

2 Related Work

In comparison with supercomputers and computational clusters, task scheduling in Desktop Grids is more complicated because of such factors as huge hardware and software heterogeneity, lack of trust, uncertainty, etc. A wide range of algorithms has been proposed in the literature to address these challenges. The most popular optimization parameters for task scheduling in Desktop Grids are the throughput [10, 11], the makespan [12, 13], the probability to obtain a correct result [10, 14], etc.

The task replication in Desktop Grids as a method to achieve optimization aims is a wide area of study [11, 14]. It often comes along with the notion of reliability or credibility of the computing nodes [10, 11, 14]. The availability patterns predictions of the computing nodes are also used to improve task scheduling in Desktop Grids [12].

Game-theoretic methods find their application for task scheduling in Desktop Grids [15, 16]. The thesis [17] presents methods for the fair distribution of resources between heterogeneous computing nodes in order to optimize throughput and average task completion times, basing on optimization methods and game theory.

The problem of drug discovery imposes additional challenges to the task scheduling in Desktop Grids. According to drug development principles, two primary characteristics of the compounds resulting set have to be tested in the laboratory: efficiency of their interaction with the target protein, and the chemical diversity [18, 19]. In general, these two objectives are in conflict: an improvement of one leads to simultaneous

deterioration of another. Hence, we have to find solutions that balance them depending on drug developers preferences.

The most conventional method to reduce input dataset for VS down to manageable size is pre-filtering of the chemical space, leaving only representative and chemically diverse compounds that promise to show desired biological activities. In many cases, the resulting library is still large, VS results are redundant and require post-processing. Compound classes that are potentially good for a specific target, but do not comply with general considerations, can be filtered out [19]. Hence, it is important to develop new efficient VS methods for large datasets.

A recent example of genetic algorithms application [20] has been published in 2015. The authors explore the chemical space by creating generations of molecules, filtering them by desirable properties, and selecting maximally diverse sets of results. The algorithm demonstrates high efficiency in selecting diverse subsets of molecules with desired properties from large databases. Apart from genetic algorithms, compound libraries can be prepared using dissimilarity analysis [21], clustering analysis [22], partitioning methods [23] etc.

The mathematical model described in this paper has been elaborated in [24].

3 Congestion Game Model

Due to variations in chemical characteristics, molecules have different chances to show high predicted binding affinity. One can expect that these chances are higher for molecules close in topology to a known ligand [25, 26]. In contrast, molecules with very large number of atoms are less likely to become hits [27]. Thus, non-overlapping subsets of molecules in the library could be ranked by their prospectivity for VS.

The estimated prospectivity can be updated in the course of VS according to interim results. At the same time, results originating from the same subset might be redundant. The idea behind this work is to explore most prospective subsets first while keeping the desired level of diversity by restricting intensity of subsets exploration.

Let us consider a computer system with m computational nodes or players $C = C_1, \ldots C_m$, and a set of computational tasks T (data) for VS processing. Each node C_i is characterized by its computational performance ops_i, which is an average number of operations performed in a time unit.

The data set T is divided into n non-overlapping blocks $T = T_1 \cup \ldots \cup T_n$ of sizes $N_1^{total}, \ldots, N_n^{total}$ such that the estimated portion of VS hits in block T_j will be p_j. We define priority of the block T_j as

$$\sigma_j = \frac{p_j}{p_1 + \ldots + p_n}, 0 \leq \sigma_j \leq 1. \quad (1)$$

The blocks with higher priority have to be chosen first for processing. We assume that all tasks in block T_j have average computational complexity θ_j, i.e., a number of operations to process one task. Each node selects exactly one block.

The nodes make their decisions at time steps $0, \tau, 2\tau, \ldots$. After a node has processed its portion of tasks, it sends the results to the server and is ready for the next

portion. Let the utility of node C_i at time step τ express the amount of useful work performed during this step. This amount depends on the number of executed tasks from the chosen block, its computational complexity, priority, and the number of other nodes who have also chosen this block.

The fewer nodes explore block T_j simultaneously, the more valuable their work is. This condition ensures diversification of the interim set of hits. Let n_j be the number of the players who have chosen block T_j at the considered step, and $\delta(n_j)$ be the congestion coefficient for the block:

$$\delta(n_j) = \frac{m + 1 - n_j}{m}. \tag{2}$$

The utility of node C_i that chooses block T_j is

$$U_{ij} = (\sigma_j + \delta(n_j)) \frac{ops_i}{\theta_j} \tau. \tag{3}$$

Therefore, at each considered time step, we have a singleton congestion game $G = (C, T, U)$, where C is the set of players (computational nodes), T is the set of data blocks of which each node selects exactly one, and U is the set of utility functions. A strategy profile is a schedule $s = (s_1, \ldots s_m)$, where the component $s_i = j$ equals to the block T_j chosen by player C_i.

Such games have been thoroughly studied in literature. The existence of at least one Nash equilibrium in pure strategies has been proven for the case of identical players [28] and identical task blocks [29]. The equilibrium situation means that no node can increase amount of its useful work by unilaterally deviating from the schedule. Moreover, better- and best-response dynamics are guaranteed to converge to equilibrium in polynomial time [29, 30].

Heterogeneity complicates the model: utility of each player depends both on its own performance, and on the task complexity of the chosen block. But due to the form of utility functions, the game G has at least one Nash equilibrium in pure strategies [31]. The Nash equilibrium situation means the best amount of useful work given the current estimates of probabilities and the available set of nodes and molecules. Further in this paper we show that the Nash equilibrium, for the most part, corresponds to the best proportion between the chemical diversity and hits number.

4 Algorithm Implementation

4.1 Desktop Grids

Desktop Grid is a form of distributed high-throughput computing system, which uses idle time of non-dedicated geographically distributed computing nodes connected over low-speed network. The computing nodes are personal desktop computers of volunteers connected over Internet (volunteer computing) or organization desktop computers connected over local area computer network (Enterprise Desktop Grid).

On the high-level, Desktop Grid has the following architecture. The main server holds a large number of tasks that are mutually independent pieces of a computationally heavy problem. Computing nodes are connected to the server. When they are idle, they request a work from the server, receive one or more tasks, and process them independently from each other. When the node finishes the processing, it reports results back to the server. The results are then stored in the database for further usage.

The server does not distribute all available tasks because of internal uncertainties present in Desktop Grid systems. These uncertainties are related to unknown availability periods of the nodes, their speed, node failures, possible computational errors, variation of tasks complexities, etc. With heterogeneous nodes and heterogeneous tasks, the scheduling system has variety of options how to assign different tasks to different nodes or the groups of nodes.

4.2 BOINC Platform

There are a number of middleware systems for Desktop Grid computing. However, the open source BOINC platform [32] is nowadays considered as de facto standard. Since 1990s, BOINC has been a framework for many independent volunteer computing projects. Today, it is the most actively developed Desktop Grid middleware, which supports the widest range of applications.

BOINC is based on server-client architecture, where the workflow proceeds as described in the previous subsection. The client part is able to work at an arbitrary number of computers with various hardware and software characteristics. The server part consists of the Web server that provides functionality to employ volunteer computing power, the database server that monitors the state of tasks and results, stores information about the clients and the whole computational process, and a set of daemon programs that periodically check the database state and implement necessary actions to operate the system and distribute the tasks.

In addition to the standard components of the BOINC server, each computational project might need individually developed utilities. One of them is a *task generator,* which creates computational tasks with specified parameters and attributes. The *client application* is developed individually for the project. This application can be either developed from scratch under the BOINC platform or be adapted using a wrapper program, which allows to run non-native applications under BOINC.

4.3 Implementation

In this section, we describe the Desktop Grid application, which implements our approach to VS in BOINC Desktop Grid middleware based on resources of the Karelian Research Center, RAS. The proposed implementation is intended for the BOINC server version 7.7.0 as well as for earlier versions.

In order to implement task scheduling algorithm proposed in this paper, one needs to implement the task generating program, modify the scheduler and assimilator daemons at the server side, and implement the docking application for the client side.

The BOINC server takes up every other part of the task workflow apart from the modifications proposed in this subsection.

Below, we describe the necessary modifications.

Firstly, the task generator must be able to create computational tasks based on the current knowledge about the input dataset structure, as defined in Sect. 3. According to the mathematical model, the knowledge about input dataset structure is being updated after each step of computations. Therefore, the task generator must consider the Desktop Grid properties as well, in order to supply the necessary amount of tasks for each subsequent step.

The task generator pseudo code is provided in Fig. 1.

```
1: for Molecule in Database
2:  Properties = get_properties (Molecule)
3:  Block = get_block (Properties)
4:  create_task (Molecule, Block)
```

Fig. 1. The task generator pseudo code.

Secondly, the scheduler must consider the current state of computational process and assign new tasks to the clients who ask for work. The assignment is being performed according to the equilibrium game schedule as described in Sect. 3.

The scheduler pseudo code is provided in Fig. 2.

```
1: if update signal received
2:  Dataset_structure = get_dataset_structure (Database)
    #s is the initial schedule vector where each client
    #chooses a separate non-empty block
3:  for i = 1 to m   #m is the number of BOINC clients
4:    s[i] = i % n   #n is the number of non-empty blocks
    #The optimal schedule vector is then calculated
    #by finite improvements sequence starting from s
    #according to the formulae (2)-(4)
5:  Schedule = get_equilibrium (s, Dataset_structure)
    #Each component of Schedule is the block number
    #chosen by the corresponding BOINC client
6: if request signal received
7:  Client = client_id
8:  Block = Schedule[Client]
9:  Timespan = requested_timespan
10: send_work (Client, Block, Timespan)
```

Fig. 2. The scheduler pseudo code.

Thirdly, the assimilator must handle completed tasks. If the result is not erroneous, it must be considered for updating the knowledge of the input dataset structure. The assimilator pseudo code is provided in Fig. 3.

```
1: if Result received
2:   if Result.error_mask != 0
3:     handle_error (Result)
4:else
5:   if Result.value >= Hit_threshold
6:     update Hits table in Database
7:   update Blocks table in Database
8:   send update signal
```

Fig. 3. The assimilator pseudo code.

Finally, the client application must perform the molecular docking. According to the BOINC system workflow for VS, the input files are the target structure, the ligand structure and the docking configuration file. The output value is the predicted binding energy. A variety of docking programs can be used as the client application for VS.

5 Experimental Setup

5.1 Database Preparation

In order to perform computational experiments and evaluate the performance of the developed approach, we divide a molecules database into blocks and simulate VS. The efficiency of the approach can be shown by earlier acquisition and higher diversity of hits at early VS stages comparing with known heuristics used in practice.

We use the database GDB-9 of about 320 thousand enumerated organic molecules with variety of chemical properties. The chosen database is manageable for performing computational experiments and can be unambiguously divided into several non-overlapping blocks. Nevertheless, the set of molecules is rich enough to demonstrate the feasibility and practicability of proposed solutions.

For the experiments, we consider three pre-calculated chemical properties of each molecule: the total number of atoms, the polar surface area *PSA*, and the partition coefficient *logP*. Basing on these properties, we divide the database into 16 non-overlapping task blocks.

As 10% of molecules in GDB-9 have $\boldsymbol{logP} \geq \boldsymbol{x} = 1.8823$, the value $\boldsymbol{x} = 1.8823$ is taken as a threshold to count a molecule as a hit.

5.2 The Chemical Diversity Measure

We employ the knowledge about input dataset structure to describe and investigate the chemical diversity of interim results. In perfect case, the fraction of hits discovered in

each pre-defined block at each computational step should be equal for all the blocks. We define the diversity of a result subset as expression (4) shows. Here, h_i is the number of hits in block T_i, p_i is the estimation of hits fraction in the block, and N_i^{total} is the initial block size.

$$D = \max_{1 \le i \le n} \frac{h_i}{p_i N_i^{total}} - \min_{1 \le i \le n} \frac{h_i}{p_i N_i^{total}}. \quad (4)$$

In Subsection 5.4 we provide the dynamics of the chemical diversity obtained at each computational step and show that the proposed scheduling algorithm outperforms two scheduling heuristics at early steps.

5.3 Experimental Setup

We performed numerical experiments, simulating homogeneous and heterogeneous Desktop Grid consisting of 64 computing nodes. As a first test case, we consider the case with identical computational nodes and tasks of identical complexities. At the second test case, we consider a Desktop Grid with heterogeneous nodes and heterogeneous tasks. The parameters of the simulations are provided in Table 1.

Table 1. Simulation parameters.

Parameter	Value		Description
	First test case	Second test case	
ops	25	15 (nodes C_1–C_{16})	Performance of a computational node (number of conditional operations per time unit)
		20 (nodes C_{17}–C_{32})	
		25 (nodes C_{33}–C_{48})	
		30 (nodes C_{49}–C_{64})	
θ	100	100 (blocks T_1–T_4)	Complexity of a computational task (number of conditional operations)
		125 (blocks T_5–T_8)	
		125 (blocks T_9–T_{12})	
		150 (blocks T_{13}–T_{16})	

At each VS step, the optimal schedule is computed based on the current knowledge about expected fractions of hits in blocks. After completion of the computations, the expected fractions of hits are updated according to the number of hits discovered in each block. Then a next step is performed, etc.

The performance of the proposed game scheduling, where each node selects a task block defined by the Nash equilibrium, is compared with two simple scheduling strategies: Probabilistic scheduling and Uniform scheduling. The probabilistic scheduling strategy represents the case when the selection of a task block does not depend on the congestion level of the block, but only on the probability to find a hit. Simulation results for the probabilistic scheduling are averaged on 20 runs. On the contrary, the uniform scheduling strategy ensures the least possible level of congestion,

or the highest diversity, by distributing the nodes across task blocks uniformly, so, as many diverse blocks are explored simultaneously as possible.

5.4 Experimental Analysis

In Figs. 4 and 5, we provide the rate of discovery hits during the first and second test cases, respectively. Each discrete point represents the fraction of the total amount of hits discovered at the corresponding step. The quality measure of such curves is their proximity to the upper left corner of the chart.

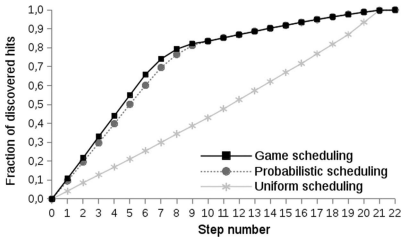

Fig. 4. Fraction of hits discovered at each step of simulations (identical nodes, identical tasks).

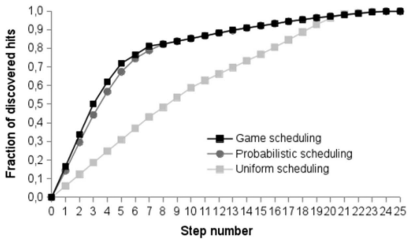

Fig. 5. Fraction of hits discovered at each step of simulations (heterogeneous nodes, heterogeneous tasks).

To illustrate the performance in terms of obtained chemical diversity, in Figs. 6 and 7, we provide the normalized chemical diversity obtained during the first and the second test cases, respectively. The chemical diversity is defined by formula (4).

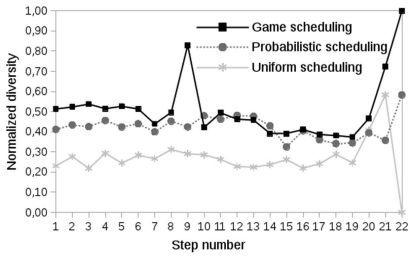

Fig. 6. Normalized diversity obtained at each step of simulations (identical nodes, identical tasks).

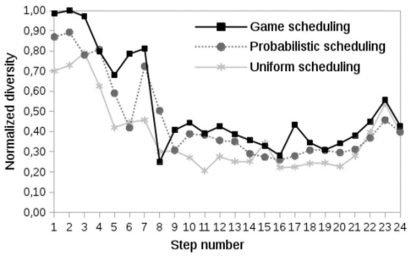

Fig. 7. Normalized diversity obtained at each step of simulations (heterogeneous nodes, heterogeneous tasks).

The presented results indicate that the game scheduling algorithm outperforms both heuristics by the fraction of discovered hits at early steps. Performance of the probabilistic algorithm is approximately equal to performance of the game scheduling

algorithm in terms of the speed of hits discovery. However, the performances in terms of obtained chemical diversity at early steps differ significantly.

6 Conclusion and Discussion

In this paper, we present an implementation of congestion game-based scheduling algorithm for high-throughput virtual drug screening using BOINC-based Desktop Grid. It is based on the mathematical model of game theory, where task scheduling is considered as a game with computing nodes as players, who choose specific data blocks for processing. We show that the equilibrium solution corresponds to the best balance between the number of interim hits and their chemical diversity. We discuss key points of implementations: the task generating program, scheduler and assimilator daemon at the server side, and present pseudo codes.

We perform computational experiments on the Enterprise Desktop Grid based on resources of the Karelian Research Center, RAS. We compare the algorithm with two known heuristics used in practice and observe that game-based scheduling outperforms them by the hits discovery rate and chemical diversity at earlier steps.

However, further study is required to assess its performance and effectiveness in multi objective domains. This will be the subject of future work. Moreover, game equilibrium solutions stability and selection of the most efficient algorithms among all equilibria are other important issues to be addressed.

Acknowledgments. This work is partially supported by the Russian Fund for Basic Research under grants no. 16-07-00622 and 15-29-07974, and CONACYT (Consejo Nacional de Ciencia y Tecnología, México) under grant no. 178415.

References

1. Pharmaceutical Research and Manufacturers of America (PhRMA). Biopharmaceutical Industry Profile (2016). http://phrma.org/sites/default/files/pdf/biopharmaceutical-industry-profile.pdf accessed 2017/05/14
2. Bielska, E., Lucas, X., Czerwoniec, A., et al.: Virtual screening strategies in drug design — methods and applications. J. Biotechnol. Comput. Biol. Bionanotechnol. **92**(3), 249–264 (2011)
3. Bohacek, R.S., McMartin, C., Guida, W.C.: The art and practice of structure-based drug design: A molecular modeling perspective. Med. Res. Rev. **16**(1), 3–50 (1996)
4. Irwin, J., et al.: ZINC: a free tool to discover chemistry for biology. J. Chem. Inf. Model. **52**, 1757–1768 (2012)
5. Bento, A.P., et al.: The ChEMBL bioactivity database: an update. Nucleic Acids Res. **42**, 1083–1090 (2014)
6. Pence, H.E., Williams, A.: ChemSpider: an online chemical information resource. J. Chem. Educ. **87**(11), 1123–1124 (2010)
7. Bolton, E.E., et al.: Chapter 12 - PubChem: integrated platform of small molecules and biological activities. Annu. Rep. Comput. Chem. **4**, 217–241 (2008). Elsevier

8. Ruddigkeit, L., van Deursen, R., Blum, L.C., Reymond, J.-L.: Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17. J. Chem. Inf. Model. **52**, 2864–2875 (2012)
9. Liu, T., et al.: Applying high performance computing in drug discovery and molecular simulation. Nat. Sci. Rev. **3**(1), 49–63 (2016)
10. Yasuda, S., Nogami, Y., Fukushi, M.: A dynamic job scheduling method for reliable and high-performance volunteer computing. In: 2nd International Conference on Information Science and Security (ICISS 2015), pp. 1–4. IEEE (2015)
11. Sonnek, J., Chandra, A., Weissman, J.: Adaptive reputation-based scheduling on unreliable distributed infrastructures. IEEE Trans. Parallel Distrib. Syst. **18**(11), 1551–1564 (2007)
12. Byun, E., et al.: MJSA: Markov job scheduler based on availability in desktop grid computing environment. Futur. Gener. Comput. Syst. **23**, 616–622 (2007)
13. Gil, J.-M., Kim, S., Lee, J.: Task scheduling scheme based on resource clustering in desktop grids. Int. J. Commun. Syst. **27**(6), 918–930 (2014)
14. Miyakoshi, Y., Watanabe, K., Fukushi, M., Nogami, Y.: A job scheduling method based on expected probability of completion of voting in volunteer computing. In: 2nd International Symposium on Computing and Networking, pp. 399–405. IEEE (2014)
15. Wang, Y., et al.: Toward integrity assurance of outsourced computing — a game theoretic perspective. Futur. Gener. Comput. Syst. **55**, 87–100 (2016)
16. Donassolo, B., et al.: Non-cooperative scheduling considered harmful in collaborative volunteer computing environments. In: Proceedings of 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 144–153 (2011)
17. Legrand, A.: Scheduling for large scale distributed computing systems: approaches and performance evaluation issues. Distrib. Parallel, Clust. Comput. [cs.DC], Université Grenoble Alpes, p. 167 (2015)
18. Tanrikulu, Y., Krüger, B., Proschak, E.: The holistic integration of virtual screening in drug discovery. Drug Discov. Today **18**(7/8), 358–364 (2013)
19. Lionta, E., Spyrou, G., Vassilatis, D.K., Cournia, Z.: Structure-based virtual screening for drug discovery: principles, applications and recent advances. Curr. Top. Med. Chem. **14**, 1923–1938 (2014)
20. Rupakheti, C., Virshup, A., Yang, W., Beratan, D.N.: Strategy to discover diverse optimal molecules in the small molecule universe. J. Chem. Inf. Model. **55**, 529–537 (2015)
21. Ashton, M., et al.: Identification of diverse database subsets using property-based and fragment-based molecular descriptions. Quant. Struct. Act. Relationsh. **21**, 598–604 (2002)
22. Downs, G.M., Barnard, J.M.: Clustering methods and their uses in computational chemistry. Rev. Comput. Chem. **18**, 1–40 (2003)
23. Oprea, T.I., Gottfries, J.: Chemography: the art of navigating in chemical space. J. Comb. Chem. **3**, 157–166 (2001)
24. Nikitina, N., Ivashko, E., Tchernykh, A.: Congestion game scheduling for virtual drug screening optimization. J. Comput. Aided Mol. Des. (2017). Manuscript submitted for publication
25. Patterson, D.E., et al.: Neighborhood behavior: a useful concept for validation of "molecular diversity" descriptors. J. Med. Chem. **39**, 3049–3059 (1996)
26. Willet, P., Barnard, J.M., Downs, G.M.: Chemical similarity searching. J. Chem. Inf. Comput. Sci. **38**(6), 983–996 (1998)
27. Hann, M.M., Leach, A.R., Harper, G.: Molecular complexity and its impact on the probability of finding leads for drug discovery. J. Chem. Inf. Comput. Sci. **41**, 856–864 (2001)
28. Rosenthal, R.: A class of games possessing pure-strategy Nash equilibria. Int. J. Game Theor. **2**(1), 65–67 (1973)

29. Milchtaich, I.: Congestion games with player-specific payoff functions. Games Econ. Behav. **13**, 111–124 (1996)
30. Ieong, S. et al.: Fast and compact: a simple class of congestion games. In: Proceedings of AIII, pp. 1–6 (2005)
31. Gairing, M., Klimm, M.: Congestion games with player-specific costs revisited. In: Vöcking, B. (ed.) SAGT 2013. LNCS, vol. 8146, pp. 98–109. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41392-6_9
32. Anderson, D.P.: BOINC: A system for public-resource computing and storage. In: Proceedings of 5th IEEE/ACM International Workshop on Grid Computing, pp. 4–10 (2004)