ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2017, Vol. 38, No. 5, pp. 926–930. © Pleiades Publishing, Ltd., 2017.

Volunteer Computing for Computational Materials Design

N. Khrapov[1*], V. Roizen[2], M. Posypkin[3], A. Samtsevich[4], and A. R. Oganov[2,4,5,6]

(Submitted by L. N. Shchur)

[1]*Institute for Information Transmission Problems, Moscow, 127051 Russia*
[2]*Moscow Institute of Physics and Technology, Dolgoprudny, Moscow oblast, 141701 Russia*
[3]*Federal Research Center "Computer Science and Control," Moscow, 119333 Russia*
[4]*Skolkovo Institute of Science and Technology, Skolkovo Innovation Center, Moscow, 143026 Russia*
[5]*Stony Brook University, Stony Brook, NY 11794, USA*
[6]*International Center for Materials Discovery, Northwestern Polytechnical University, Xi'an, 710072, China*

Received December 21, 2016

Abstract—The problem of crystal structure prediction is very old and does, in fact, constitute the central problem of theoretical crystal chemistry. In this paper, we discuss the popular USPEX evolutionary algorithm for crystal structure prediction. Here we present the distributed computing implementation of USPEX based on a popular BOINC volunteer computing platform, and discuss experimental results and project performance.

DOI: 10.1134/S1995080217050195

Keywords and phrases: *distributed computing, evolutionary algorithm.*

1. INTRODUCTION

Computational materials design is a revolutionary field of science. While experimental materials discovery even today is mostly based on trial-and-error and serendipity, it has become possible to predict new materials on the computer [1]. State-of-the-art computational methods can be applied to prediction of materials for large variety of applications: e.g. thermoelectric materials dielectrics, superhard materials, magnets etc. However, large-scale application of such techniques requires vast amount of computational resources. One of the most powerful methods for computational materials design is the USPEX evolutionary algorithm [2]. Such reputation was confirmed by blind tests of Organic and Inorganic Crystal Structure Prediction [1], as well as by numerous studies [1, 3]. In addition, USPEX has been successfully used for prediction of structure of nanoparticles [4] and polymers [5].

Here we present the distributed computing implementation of USPEX based on a popular BOINC volunteer computing platform [6]. Volunteer computing is a powerful tool which has been successfully applied for protein structure prediction; e.g., in the Rosetta@HOME project [7]. Problems of computational materials design and ab initio protein structure prediction are similar complexity optimization problems. That is why application of the distiributed computing seems as a promising approach.

Our implementation works as follows. USPEX server runs USPEX core application that generates tasks. The project server downloads USPEX tasks from the USPEX server and submits them to the BOINC infrastructure. Volunteer computing nodes run the external code(s) for local optimization of structures—here we use GULP [8], with the received tasks as inputs. After the tasks are computed the results are collected by the BOINC server. Then obtained results are downloaded and processed by the USPEX server. In the talk we present experimental results and discuss project performance.

*E-mail: nkhrapov@gmail.com

2. COMPUTATIONAL MATERIALS DESIGN

Rational materials design is based on the fact that their stability and physical properties are determined by their structures and chemical composition. By selecting the correct configuration of atoms, one can obtain a compound with the desired properties. From the mathematical point of view, this is a multi-parameter optimization problem, which can be efficiently solved by computational techniques.

The development of the robust optimization method is crucial here. Let us consider as an example crystal structure prediction. By simple combinatorial argument [2] the number of possible distinct structures can be evaluated as $C = \binom{V/\beta^3}{N} \prod_i \binom{N}{n_i}$, where N is the total number of atoms in the unit cell of volume V, β is a reasonable discretization parameter of order of 1 A. For systems of even moderate complexity, C is a huge number.

That is why it is important to design a search algorithm based on the understanding of the specifics of the studying systems: crystal, nanoparticles or polymers. However, even state-of-art techniques require a vast amount of computational resources especially for solution of complex problems. This makes necessary the development of modern computational systems adapted to the materials design.

3. EVOLUTIONARY ALGORITHM

USPEX is an evolutionary algorithm [2] originally developed for crystal structure prediction, though its current version supports as well prediction of the nanoclusters [4] and polymers [5]. At the heart of its success is the combination of global and local optimization, allowing one to explore the potential energy surface efficiently and avoid “getting stuck” in local minima. The general scheme of the USPEX workflow is presented in Fig. 1. Let us provide below a brief explanation of it.

Calculation is considered as a sequence of generations consisting of structures. The first generation is created randomly using the symmetry groups relevant for the problem (e.g. space groups for crystal). Then all structures are accurately relaxed to find local minima. For relaxation procedure USPEX is interfaced with a number of programs performing quantum-chemical calculations or classical calculations based on force fields. In this work General Utility Lattice Program (GULP) [8] was used. This is a free code, which allows to conduct relaxation using a number of force fields.

After relaxation, a fixed percentage of the fittest structures are selected to produce the next generation by several evolutionary operators. There are several implemented fitness function in the USPEX code. Details of work of evolutionary variation operators was described in other publications [2, 9]. Finally, a certain amount of random structures are added to generation, to maintain a diversity of the population, crucial for the evolutionary algorithm. After that the whole procedure is repeated. The halting criterion is that the fittest structure remains unchanged for a certain number of generations.

The main part of computational resources is spent on relaxation. Implementation of the volunteer computing provides an opportunity to speed up the calculation and significantly expand the complexity of problems explored. Below we present details of our developments.

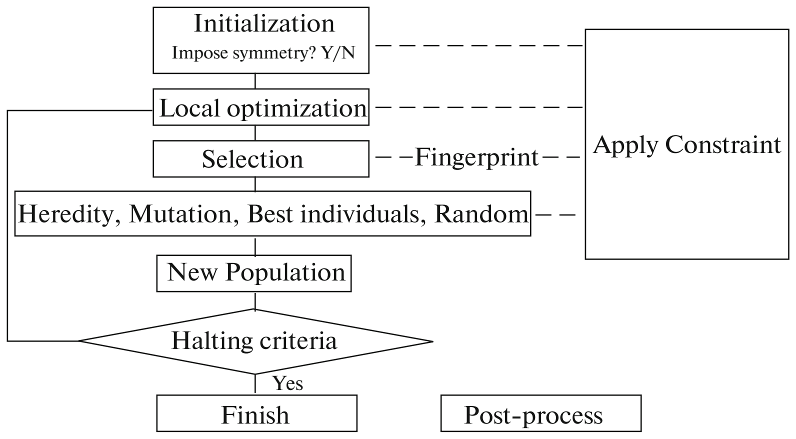

Fig. 1. Illustration of the USPEX algorithm.

4. VOLUNTEER COMPUTING

The term "volunteer computing" stands for a system comprising a set of personal computers distributed over the world. Technically a volunteer computing project consists of the server and client PCs connected to it. Typically a client PC belongs to a person who wishes to donate the unused power of their computer for a project interesting to them. This explains the word "volunteer."

The technical features of volunteer computing are as follows: the only allowed type of communications is the information exchange between a client PC and a server. There are no direct communications among the computational nodes (clients); the initialization time is large. This time is necessary for job preparation and data transfer. The job preparation takes about 20 seconds. Data transfer time depends on the size of executable files and data files; computational nodes are unstable in a sense that they can leave or join computations at virtually any moment, because a volunteer periodically turns his PC off.

Presently the most widely adopted software for volunteer computing is BOINC (Berkeley Open Infrastructure for Network Computing) [6]. It is divided into two parts: BOINC-client and BOINC-server. The BOINC-client is installed on client PCs and provides the following functionality: connects to BOINC projects; requests and downloads tasks from the project's server; runs tasks on the volunteer computer; uploads output files to the project's server.

The server part of the BOINC-infrastructure is installed on a project's server. It does the following: generates work units (jobs); responds to client requests; receives and processes results from the client's part; runs web-server for the project information.

A distributed application is deployed at the project's server and is divided into two parts: client part and server part. The client part is a solver for the problem under consideration. It performs the computations. The server part's responsibility include the following: preparing input files for client side; processing results obtained from nodes; finalizing the computations based on the data obtained from client PCs.

5. INTEGRATING USPEX AND BOINC

The client-server organization of the USPEX tool perfectly suits for volunteer computing. USPEX provides the ability to add integration interfaces thereby making integration with BOINC quite natural and convenient.

The problem is that the running time of USPEX's job varies from 10 seconds to a few hours and is not known in advance. Thus generating BOINC work unites from one USPEX job is not a wise decision because there is a high probability that some work-units will run almost zero time and packaging expenses will kill the performance. Therefore, several USPEX's jobs are aggregated to one BOINC-task. A client part of a distributed application is a script running multiple serial USPEX-tasks.

We developed a software component that performs the USPEX-job submission to the BOINC-infrastructure and the results delivery. This component consists of a plugin for USPEX-server and a plugin for the BOINC server. The former collects USPEX jobs into one folder on USPEX server, packages them sends them to the BOINC server in a single scp-session. Periodically, the USPEX plugin queries the status of jobs and delivers results of completed jobs to the USPEX server. BOINC plugin receives jobs from USPEX plugin, and then submits work-units to the BOINC computational infrastructure. Then BOINC plugin removes completed tasks from the infrastructure and sends results to the USPEX-plugin.

6. DISCUSSION

In order to test stability of the system, several calculations have been conducted. Here we present only results of the standrad USPEX example of the prediction of $MgAl_2O_4$ at the pressure of 100 GPa pressure. This is a variable-cell calculation using Buckingham potentials, using the GULP code. This example has direct bearing on the physics of the Earth's interior.

The number of atoms was set to be 28 in the primitive cell. The first generation of structures was produced using the random symmetric algorithm; all subsequent generations were produced using the following evalutionary operators: heredity (initial fraction 50%), permutation (initial fraction 10%), softmutation (initial fraction 20%) and random symmetric generator (initial fraction 20%). In subsequent generations, fractions of these variation operators were adjusted on-the-fly by USPEX.

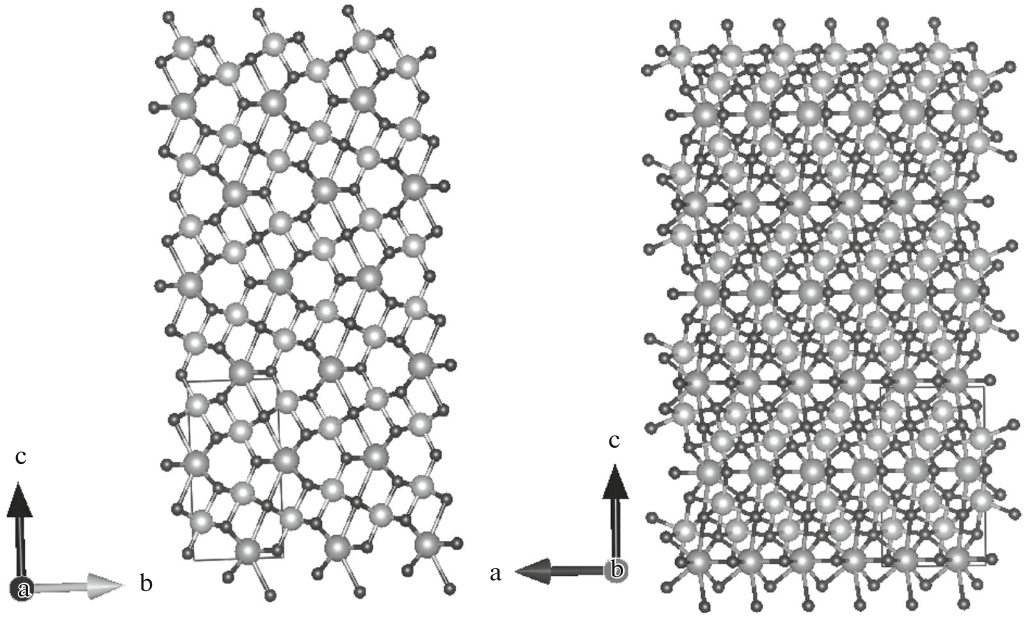

Fig. 2. Stable crystal structure of $MgAl_2O_4$ at the pressure of 100 GPa, predicted by USPEX and shown in different projections. Space group: Cmcm; lattice parameters: $a = 2.63$ Å, $b = 8.78$ Å, $c = 8.72$ Å; Mg(0.00, 0.12, 0.75), Si(0.50, 0.13, 0.076), O(0.50, 0.05, 0.25), O(0.00, 0.27, 0.11), O(0.00, 0.00, 0.00).

The same calculation (because of the stochastic nature of USPEX algorithm) was run 5 times. In all cases we have obtained the reference structure in the first 5 generations. Predicted structure is presented in Fig. 2.

It is important to raise the question about adaptation of the USPEX algorithm for volunteer computing. Usual calculations are performed with a small number of structures in a population. Quality of achieved results is due to the stability and efficiency of the USPEX algorithm. However, with the tremendous resources of distributed computing it is possible to conduct a high-throughput computational screening for new materials.

On the other hand, USPEX is facing two main obstacles. The first one is connected to the fact, that originally USPEX was developed for calculations running on clusters and supercomputers. Such systems have a stable architecture and fast data exchange rate. This is crucial for the evolutionary algorithm, which can not produce a new generation without full completion of the current one.

The other one is not strongly related to USPEX itself. The most accurate and reliable calculations require quantum chemistry calculations, which are extremely time consuming. Our goal is to enable stable distributed computing with these codes.

ACKNOWLEDGMENTS

A.R. Oganov acknowledges the support of the Russian Science Foundation (grant no. 16-13-10459) for financial support. M. Posypkin and N. Khrapov acknowledge the support of the Russian Foundation for Basic Research (project nos. 17-07-00510, 16-07-00873, 16-07-00814, and 16-07-00458) and of Presidium of the Russian Academy of Sciences (programs I.33 and I.5).

REFERENCES

1. *Modern Methods of Crystal Structure Prediction,* Ed. by A. R. Oganov (Wiley-VCH, Berlin, 2010).
2. A. R. Oganov and C. W. Glass, "Crystal structure prediction using ab initio evolutionary techniques: principles and applications," J. Chem. Phys. **124**, 244704 (2006).
3. W. W. Zhang, A. R. Oganov, A. F. Goncharov, Q. Zhu, S. E. Boulfelfel, A. O. Lyakhov, E. Stavrou, M. Somayazulu, V. B. Prakapenka, and Z. Konopkova, "Unexpected stoichiometries of stable sodium chlorides," Science **342**, 1502–1505 (2013).

4. N. L. Matsko, E. V. Tikhonov, V. S. Baturin, S. V. Lepeshkin, and A. R. Oganov, "The impact of electron correlations on the energetics and stability of silicon nanoclusters," J. Chem. Phys. **145**, 074313 (2016).
5. V. Sharma, C. Wang, R. G. Lorenzini, R. Ma, Q. Zhu, D. W. Sinkovits, G. Pilania, A. R. Oganov, S. Kumar, G. A. Sotzing, S. A. Boggs, and R. Ramprasad, "Rational design of all organic polymer dielectrics," Nat. Commun. **5**, 4845 (2014).
6. D. P. Anderson, "Boinc: A system for public-resource computing and storage," in *Grid Computing, Proceedings of the 5th IEEE/ACM International Workshop, Nov. 8, 2004* (IEEE, Washington, DC, 2004), pp. 4–10.
7. K. Simons, C. Kooperberg, E. Huang, and D. Baker, "Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions," J. Mol. Biol. **268**, 209–225 (1997).
8. J. D. Gale and A. L. Rohl, "The general utility lattice program (GULP)," Mol. Simul. **29**, 291–341 (2003).
9. A. R. Oganov, A. O. Lyakhov, and M. Valle, "How evolutionary crystal structure prediction works and why," Accounts Chem. Res. **44**, 227–237 (2011).