ISSN 1063-7796, Physics of Particles and Nuclei, 2026, Vol. 57, No. 4, pp. 587–589. © Pleiades Publishing, Ltd., 2026.

Experience of Applying Volunteer Computing to Solve Scientific Problems Using an Evolutionary Algorithm

N. P. Khrapov[a], *, A. R. Oganov[b], and M. G. Kostenko[b]

[a] *Institute for Information Transmission Problems, Rissian Academy of Sciences (Kharkevich Institute), Moscow, Russia*

[b] *Skolkovo Institute of Science and Technology, Moscow, Russia*

**e-mail: nkhrapov@gmail.com*

Received September 17, 2025; revised November 10, 2025; accepted December 1, 2025

Abstract—This paper proposes practical methods for improving performance when using an evolutionary algorithm in a volunteer computing environment. The analysis showed that the low stability of some nodes leads to a delay in the creation of a new generation. The paper will propose a way to overcome this problem by maintaining a stable resource.The problem of the inability to pre-estimate task durations is also discussed and methods for overcoming it are proposed.

DOI: 10.1134/S1063779626700206

INTRODUCTION

Evolutionary algorithm (hereinafter referred to as EA) is a heuristic optimization method widely applicable in various fields of science and technology. A description of the use of EA for predicting the structure of matter is given in [1]. A description of the computing complex integrating the USPEX and BOINC systems is given in [2]. When solving problems using EA on desktopgrid, a number of specific difficulties arise that reduce efficiency and productivity.

Let us consider the impact of low stability of computing nodes on the efficiency and productivity of EA. Some nodes download a computing task, but for a number of reasons do not return the result in a reasonable time. This is the reason for the delay in creating a new generation based on all the results of the previous generation. A detailed analysis of this cause of productivity loss is given in [3] as the "one person does not delay a wedding" problem. This publication proposes a way to overcome this problem by supporting a stable resource based on a queuing system. The paper describes practical aspects of building a hybrid system integrating BOINC, USPEX and SLURM [4]. Various aspects of making a decision on the resource to be used are considered.

Now let us consider the problem of reduced efficiency due to the fundamental impossibility of preliminary assessment of the computational complexity of a task. Modern methods allow packing several small tasks into a single BOINC task for sequential execution on a computing node. Without preliminary estimation, the execution time of some composite BOINC-tasks will exceed reasonable limits. In the absence of a packing mechanism, individual tasks will be executed excessively quickly, which is inefficient with high initialization overhead. The article proposes universal methods for managing the computational complexity of BOINC tasks based on time control on the side of the computing node.

USING QUEUE SYSTEM SUPPORT

The support of the queue system is based on the principle that copies of lost tasks will be executed on a stable computing resource (queue system). Integration of systems allows the possibility of using typical scenarios for interaction between the queue system and USPEX. Within the USPEX@HOME project, SLURM is used as a supporting queue system.

Let's consider the problem of deciding whether to use BOINC or SLURM. The decision-making mechanism is implemented as separate functions *reservedecider* and *taskreservedecider*. It is assumed that the user can configure these functions to take into account the specifics of the infrastructure and the tasks being solved.

The *reservedecider* function decides whether to use SLURM for all remaining tasks. This function is called after each cyclic launch of USPEX once for all tasks. The main criterion for deciding whether to use the reserve resource is the ratio of the number of remaining structures N_{remain} to the specified threshold value N_{limit}. If $N_{\text{remain}} < N_{\text{limit}}$, then copies of all tasks previously sent to BOINC will be submitted to SLURM. Also, all new tasks generated by USPEX will be sent to SLURM, but not sent to BOINC. This approach

allows us to overcome the "one person does not delay the wedding" problem. Tasks that delay the creation of a new generation will be executed on a stable computing resource.

The *taskreservedecider* function is run for each new task if reservedecider returned a negative value. The taskreservedecider function is applicable in the following cases:

• it is necessary to take into account the characteristics of the task (for example, the relaxation step);

• sending tasks to SLURM if the queue system has a low load;

• the ability to send a task to SLURM if the waiting time for the result from BOINC exceeds the specified $T_{\text{taskdeadline}}$ value. Using the $T_{\text{taskdeadline}}$ parameter allows you to reduce the execution time of the entire generation. Let's say a task was lost on a compute node at the first relaxation step. Then, when the reservedecider function is triggered, all relaxation steps will be executed on the reserve resource, starting from the first one. This can take a long time and delay the generation of a new generation. When using $T_{\text{taskdeadline}}$, copies of tasks lost at early relaxation steps will be sent to SLURM before N_{limit} is triggered. Therefore, when N_{limit} is triggered, the remaining tasks will start executing from later relaxation steps;

• using the $N_{\text{softlimit}}$ parameter. If $N_{\text{remain}} < N_{\text{softlimit}}$, then subsequent tasks generated by USPEX will be sent to SLURM. But tasks previously sent to BOINC will not be redirected to SLURM. Using two parameters N_{limit} and $N_{\text{softlimit}}$ allows avoiding duplicate execution of identical tasks in BOINC and SLURM. If you use only N_{limit}, then when the threshold value is reached, copies of tasks recently sent to BOINC will be sent to SLURM.

When executing EA, there are always periods of generation completion when the task does not use the main resources of the infrastructure. This leads to a decrease in efficiency and the transition of the BOINC to the waiting state [5]. To avoid this problem, it is advisable to solve several fundamental tasks simultaneously. In this way, the infrastructure will perform the tasks of one task when another task is in the generation completion state.

CONTROL OF COMPUTATIONAL COMPLEXITY OF BOINC-TASKS

Computing practice has shown that the duration of a single task generated by USPEX can range from 1 min ($T_{\text{USPEXmin}} = 1$ min) and 10 h ($T_{\text{USPEXmax}} = 600$ min). The type of the probability density function of the execution time depends on the specifics of the problem being solved and also cannot be estimated in advance. Therefore, a universal approach is needed for various configurations of the task execution time distribution density.

Suppose we pack 5 USPEX tasks into one BOINC task. Then $T_{\text{BOINCmin}} = A \cdot T_{\text{USPEXmin}} = 5$ min and $T_{\text{BOINCmax}} = A \cdot T_{\text{USPEXmax}} = 3000$ min $= 50$ h. Here A is the packing parameter (the number of USPEX tasks in one BOINC task). Tasks longer than 15 h are undesirable. Firstly, excessive execution time delays the creation of a new generation of EAs. Secondly, excessively long tasks increase the probability of their loss on the side of the computing node.

Within the proposed approach, the time at the very beginning of the calculations is recorded on the computing node side. Before starting the next USPEX task, the elapsed time since the beginning of the calculations T_{current} is checked. If $T_{\text{current}} > T_{\text{limit}}$, then the execution of this BOINC task is completed, and the USPEX results that have been completed by this time are returned to the server. The USPEX tasks from the unfinished part of the BOINC task will be resubmitted either to BOINC or to SLURM, depending on the state of the entire generation's computation. Let us estimate the maximum execution time for the BOINC task. Then $T_{\text{BOINCmax}} = T_{\text{USPEXmax}} + T_{\text{limit}}$. In the previously considered example if $T_{\text{limit}} = 2$ h, then $T_{\text{BOINCmax}} = 10 + 2 = 12$ h. Such execution time is acceptable for the tasks being solved.

At present, parameters A, N_{limit}, $N_{\text{softlimit}}$, $T_{\text{taskdeadline}}$, T_{limit} are selected empirically in the process of infrastructure control. The development of a methodology for optimal selection of parameters is the subject of further research.

FUNDING

The research was carried out within the state assignment of Ministry of Science and Higher Education of the Russian Federation for the Institute for Information Transmission Problems, Rissian Academy of Sciences (no. 125031803938-5).

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. A. R. Oganov, Ch. J. Pickard, Q. Zhu, and R. J. Needs, "Structure prediction drives materials discovery," Nat. Rev. Mater. **4**, 331–348 (2019). https://doi.org/10.1038/s41578-019-0101-8
2. N. P. Khrapov, V. V. Rozen, A. I. Samtsevich, M. A. Posypkin, V. A. Sukhomlin, and A. R. Oganov, "Using virtualization to protect the proprietary materi-

al science applications in Volunteer Computing," Open Eng. **8**, 57–60 (2018). https://doi.org/10.1515/eng-2018-0009

3. N. P. Khrapov, "Metrics of efficiency and performance when using evolutionary algorithm on Desktop Grids," Program. Comput. Software **47**, 882–886 (2021). https://doi.org/10.1134/s0361768821080041
4. M. A. Jette and T. Wickberg, "Architecture of the Slurm workload manager," in *Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, Vol. 14283 (Springer, Cham, 2023), pp. 3–23. https://doi.org/10.1007/978-3-031-43943-8_1
5. N. P. Khrapov, "Probabilistic models of the behavior of the BOINC infrastructure in typical situations," in *Supercomputing*, Lecture Notes in Computer Science, Vol. 15407 (Springer, Cham, 2024), pp. 882–886. https://doi.org/10.1007/978-3-031-78462-08