



I. A. Chernov, N. N. Nikitina

## Effective scanning of parameter space in a Desktop Grid for identification of a hydride decomposition model

**ABSTRACT.** We consider parameter identification of a hydride decomposition model by scanning the parameter space in parallel. Such problem is resource demanding, but suits best for Desktop Grid computing. Considering task retrieval as a game, we show that the search process can be improved to produce solutions faster in comparison with random search, with no or minor additional cost.

*Key words and phrases:* Desktop Grid, task scheduling, BOINC, Enterprise Desktop Grid, high-throughput computing, distributed computing, modeling of hydride decomposition.

2010 *Mathematics Subject Classification:* 68W10; 68W15, 91A80

### 1. Introduction

Inverse problems of parameter identification using experimental data are important in material science and physical chemistry. The model's ability to reproduce observations is the main criteria of its quality. A model usually contains multiple parameters which have physical sense. Nonlinear interaction of different processes makes independent determination of parameters by superposition techniques hard or even impossible. The common approach of choosing a single limiting reaction can only be used when this reaction is significantly slower than all others, which is seldom the case of close-to-equilibrium processes. A good model should fit not only a single experimental curve, but a series of them, even better if parameter sets are similar. Constructed on basic principles (e.g., conservation laws) and descriptions of elementary reactions, a model usually contains more than one parameter.


---

Financially supported by Russian Foundation for Basic Research 18-07-00628-A, 16-07-00622-A.

© I. A. CHERNOV, N. N. NIKITINA, 2018

© INSTITUTE OF APPLIED MATHEMATICAL RESEARCH KARRC OF RAS, 2018

© PROGRAM SYSTEMS: THEORY AND APPLICATIONS (DESIGN), 2018

 10.25209/2079-3316-2018-9-4-53-68



A model is a function  $M$  that maps a set of parameters (domain  $D$  from finite-dimensional space  $R^k$ ) into a curve (a continuous function or, in case of numerical model, an array). In material science, physical phenomena are described by ordinary or partial differential equations. Some function of the phase variables is the measured quantity as a function of time.

Evaluation of this function  $M$  in a given point of the domain  $D$  is the direct problem. The inverse problem, also called parameter identification, is the search for points from  $D$  that approximate given experimental curves. If the curves are compared in the  $L_2$  space, this is the least squares method, though other appropriate metric spaces can be used. A set of parameters  $s \in R^k$  that provides approximation of the given curve  $f(t)$  with the chosen precision  $\epsilon$ , i.e.,

$$(1) \quad F(s) = \|M(s) - f\| \leq \epsilon,$$

is called a solution to the inverse problem. Precision  $\epsilon$  is also called the threshold.

There are no general methods for inverse problems. They are usually non-linear even if the direct problem is linear. The reason is at least non-linearity of solution to even linear differential equations. Also, such problems are usually ill-posed [1, 2]. Existing methods provide, at best, local convergence; therefore, existence of alternative solutions remains an open question. Various attempts to provide uniqueness of the solution demand, in practice, knowledge of the solution.

The quality of the solution is measured by its ability to fit a series of curves with reasonable precision. Also, the solution should be physically sound, i.e., to accord general physical common sense and the basic properties. Obvious inequalities are set for probabilities or fractions of some quantity. Sometimes physical estimations are not strict; for example, it is known that some speed is much lower than the sound speed but no more precise estimations are available. Such mild estimations make it necessary to check potential solutions that are *a priori* unlikely to occur.

The domain  $D$  of admissible values of parameters can contain much more than a single solution to the inverse problem. If the dimensionality  $k$  of the space of parameters is not too high, blind search (also called scanning) on a mesh inside  $D$  can be fruitful. Nodes with best solutions can serve as initial approximations for algorithms of the decent type or similar ones.

This problem suits well to distributed computing systems [3, 4], including Desktop Grids, due to the following properties.

- (1) Separate tasks (single solutions of the direct problem, i.e., evaluation of  $M$  in a mesh node, and comparison of two curves) are completely independent, can be done in parallel, and do not need any synchronization.
- (2) The tasks require moderate amounts of processing power, RAM and disc space.
- (3) The volume of data necessary for solving a task is low: a  $k$ -dimensional array of values and experimental curves (sent once).
- (4) A result is a single real number (the fitting quality), so network connection does not need to be fast.

The parameter identification is generally a highly scalable problem. The size and dimensionality of the parameters domain  $D$  can be large, and variation of either precision or the grid size can make the search process even more time- and resource-demanding. At the same time, interim results obtained at the early stages of the search can significantly contribute to the process due to new knowledge about the parameters domain or justification of the model itself. This motivates one to employ sophisticated search algorithms rather than the blind search.

In [5] we described the solution of parameter identification of an aluminum hydride decomposition model by scanning the parameter space. We found much more solutions than was expected, present in all parts of the domain. However, the search process itself can be optimized.

The rest of this paper is organized as follows. First we describe the model and the inverse problem to be solved. Then we discuss BOINC-based Desktop Grids as a tool for solution and show that the policy of task request can be subject to optimization. We compare three scheduling strategies intended to produce good results faster. One of them is a game between the computing nodes so that they choose subsets of  $D$ . Improvement is achieved without centralized control which is good for a client-server architecture because no additional load on the server is necessary.

## 2. Metal hydrides decomposition

Hydrides of metals are considered as a possible solution of hydrogen accumulation and transport problem, in particular on board of hydrogen vehicles. Light metals with high weight percent of hydrogen in hydrides, like aluminum or magnesium, are of the most interest. Beside the desired large amount, the quick evolution of fuel is necessary for vehicle applications as well. This demands detailed understanding of hydrogen release from

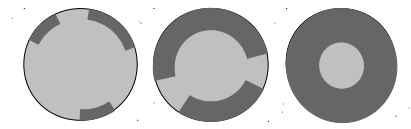


FIGURE 1. Skin formation and shrinking core scenario; light gray is for the hydride phase.

hydrides of metals and evaluation of kinetic parameters of different carrying materials.

We do not discuss modelling of decomposition of metal hydrides here: see [6, 7]. Note only, that high rates of decomposition and high amounts of accumulated hydrogen make simple approaches, like Avrami-Erofeev method, consideration of parts of kinetic curves (e.g., the almost linear part), selecting a single limiting process, quite questionable. Complexity of the model must be comparable to information provided by information data. Besides, a model must, at least, be conservative and describe concurrency between hydride decomposition and hydrogen desorption.

The model used for this work is described in [8], see also references therein. We model a single spherical particle of hydride powder, with unit radius. The chosen shrinking core scenario means that the new metal phase first forms a skin on the surface of the particle and then propagates toward centre. The thickness of this skin is  $h = 1 - \rho(t)$  where  $\rho$  is the size of the old phase core, Figure 2. The formed skin consists of symmetric nuclei completely described by their total area with respect to total area of the particle  $S$  and thickness  $h$ .

Physical assumptions are as follows:

- diffusion of hydrogen dissolved in metal is quick in small powder particles, so gradients of concentration are negligible;
- there is no re-adsorption of hydrogen, i.e., it desorbs to vacuum;
- desorption from the metal phase is through pores and cracks;
- hydrogen concentration in metal is low compared to unit concentration in hydride; however, desorption flux is not zero;
- particle size does not change during reaction;
- fraction  $\nu$  of hydrogen flux from the metal phase is due to shrinking core (the rest is due to skin expansion).

Denoting hydrogen flux densities for two phases by  $J'$  (for hydride) and  $J$  (for metal), let us write down the total hydrogen flux:

$$J'(1 - S(t)) \cdot 4\pi + JS(t)4\pi\rho^2.$$

Local conservation gives the equation for the core size:

$$(2) \quad \frac{d\rho}{dt} = -\nu(S)J, \quad \rho > 0.$$

Here  $\nu(S) = \nu$  if  $S < 1$ , i.e., if the skin can expand. If it is already ready,  $\nu(1) = 1$ .

Global conservation law gives the second equation:

$$(3) \quad \frac{dS}{dt} \frac{1 - \rho^3}{3} = J'(1 - S(t)) + (1 - \nu)JS(t)\rho^2.$$

The measured quantity was the volume of the reacted fraction, i.e., the volume of the new phase relative to the volume of the particle:

$$f(t) = (1 - \rho^3(t))S(t).$$

The model (2)–(3) contains five parameters:

- (1) hydrogen flux densities for the hydride and the metal phase  $J'$  and  $J$ ;
- (2) initial values of phase variables  $S_0$  and  $\rho_0$ ;
- (3) the shrinking core rate  $\nu$ .

All parameters are non-negative; also,  $S_0 \leq 1$ ,  $\rho_0 \leq 1$ ,  $\nu \leq 1$ . Flux densities can not be too high and therefore are softly estimated.

Therefore we have a 5-dimensional box to look for the parameters. The  $L_2$  norm of the difference between curves was expressed in terms of  $L_2$  norm of the data and so was dimensionless. We omit some technical details not necessary here.

Although the equations are simple and can be integrated analytically, numerical solution is more efficient, much more flexible and allows easy modification of the code if the model demands to be changed.

The numerical predictor-corrector method (Heun's method) implemented in standard Fortran-90 can be used on most platforms.

### 3. Computing environment and numerical experiments

Initially, the threshold for selecting good results (1) was set to a reasonable value 6%. We manually found a set of parameters that provided good fitting (6%) as a reference solution:  $\bar{J} = \bar{J}' = 10$ ,  $\bar{\rho}_0 = 88$ ,  $\bar{S}_0 = 2$ ,

and  $\bar{\nu} = 50$  (flux values are in convenient abstract units, other parameters are in %). The grid was uniform along each axis with 5 nodes for  $S_0$  (0–4) and 10 nodes for each of other parameters: 2–20 for flux densities, 80–98 for  $R_0$ , and 0–90 for  $\nu$ . So, the grid consisted of 50 thousand points which produced 250 thousand tasks for five experimental curves.

For a while, we consider different curves as separate experiments and do not prefer parameter sets that are similar for different curves; first we want to see if there are many solutions to choose from.

Numerical scanning of the parameter domain was performed on a BOINC-based [10] local Desktop Grid of Karelian Research Center<sup>1</sup>. Our aim was not only to find kinetic parameters for decomposition of the hydride of aluminum, but also to study the opportunities of employing Desktop Grids to solve this kind of problems and to find ways of improving the search process, so that high-dimensional problems would be solved quicker and at lower costs.

BOINC is one of the most popular open source middleware for Desktop Grid systems, either for volunteer computing or Enterprise Desktop Grid [11]. Such system exploit idle power of desktop computers and other non-dedicated computing resources (like servers and even mobile devices [12]).

Despite relatively low average performance of each computing node, Desktop Grids are able to collect much power per unit cost due to employing many computers at no or low additional expenses. An obvious drawback is a relatively slow communication, so that data exchange, task dependencies and synchronization spoil performance drastically. However, sets of independent tasks (bags of tasks), as in our case, suit Enterprise Desktop Grids perfectly. The BOINC platform allows to solve bags of tasks in heterogeneous environments. Applications do not need to be specially developed for the platform, nor to be open source.

BOINC has the two-level client-server architecture: the server creates the tasks; client nodes that have idle resources ask for tasks and receive them. After solving a task, a node returns the answer to the server.

During the experiments described in [5], we used one desktop computer, two servers, the control node and three computing nodes of the cluster of Karelian Research Centre (each node of the cluster has two 4-core CPUs).

---

<sup>1</sup><http://cluster.krc.karelia.ru/>

TABLE 1. Distribution of the solutions over the parameter grid;  
(a): *good* (%), (b): *best* (%).

$J$	(a)	(b)	$J'$	(a)	(b)	$\rho_0$	(a)	(b)	$\nu$	(a)	(b)	$S_0$	(a)	(b)
2	17.5	92.6	2	7.4	7.49	80	9.4	14.0	10	13.8	29.4	0	19.6	22.8
4	13.7	7.25	4	3.8	2.64	82	9.6	13.6	20	13.5	26.0	1	19.8	25.0
6	11.0	0.08	6	3.5	2.72	84	9.9	12.7	30	13.2	21.0	2	20.1	21.0
8	9.2	0.04	8	8.9	4.02	86	10.2	11.8	40	12.4	14.2	3	20.2	19.0
10	8.4	0	10	11.7	7.81	88	10.5	10.9	50	10.6	6.76	4	20.3	12.2
12	8.3	0	12	12.9	11.4	90	10.7	9.66	60	8.0	1.02			
14	8.2	0	14	13.4	14.9	92	10.8	8.52	70	6.1	0.51			
16	8.1	0	16	13.2	16.2	94	10.6	7.69	80	6.7	0.39			
18	8.0	0	18	12.9	16.9	96	9.9	6.46	90	7.8	0.44			
20	7.6	0	20	12.3	15.9	98	8.4	4.73	100	7.9	0.47			

The total time of scanning was about one hour, which was more than 23 times faster compared to one personal computer. The scanning found much more solutions than was expected. 85 988 results were better than the pre-defined threshold 6%. Therefore we reduced the threshold to 1.6% to get more than 2 537 solutions.

Note that further reduction of the quality threshold is impossible on the given dataset because some experimental curves lack solutions, though other have many. For example, reduction of the threshold to 1% gives 220 solutions, all for the same experimental curve. This critical value is unknown beforehand, so post-processing of the results is very important, and it may be necessary to repeat the scanning. Also note that locally converging methods can fail to find better solution at all, producing a locally minimal 6% solution.

In the following part of the paper, let us call the results selected by the 6% threshold the *good* ones, and the results selected by the 1.6% the *best* ones. We will use these two notions to evaluate and discuss task scheduling policies.

Table 1 illustrates the distribution of physically distinguishable solutions over the parameter grid.

#### 4. Search methods

Optimization criteria of scanning the parameter domain can be different. We used a fixed grid and needed to check every point; scanning all domain (or even a set of all possible values of parameters) seems more informative,

but hard to implement. However, a grid of variable step can be considered. The ideas presented in this paper can be used for these purposes also, but let us focus on the fixed-grid search first. Each method must check each grid point by the end of the computation. As only the order of checked points (tasks) can be chosen, optimization does not change the computation makespan; therefore we can consider only the linear order of the points, assuming one time unit necessary for one task.

To compare different ordering policies, we estimate the time needed to produce all *good* (or *best*) results or some fixed part of them.

*Native scheduling.* We used a simple residual coding of the parameter set grid to generate tasks. Each set (and therefore each task) was associated with a natural number  $n \leq 250\,000$  which produced the number of the experimental curve and one-dimensional indices of individual grids of each parameter as residues (plus 1) of division on the number of curves (5) and the sizes of individual grids (10 four times and then 5). This approach changes  $J$  faster than other parameters, and Table 1 shows that much work was wasteful because all high-quality *best* solutions belong to the domain of low values of  $J$ .

The similar situation is with  $\nu$ , but luckily this parameter is changed more slowly than the others (except  $S_0$ ). On the other hand, *good* parameters are distributed rather homogeneously (Table 1), so the search order is not significant. The largest sequential numbers of the *good* and the *best* solutions were, respectively, 249 970 and 242 535 out of 250 000. So, almost all grid points needed to be checked prior to collecting all *good* and *best* results. 90% of *good* and *best* results were obtained after computing task 246 531 and 96 534 respectively, which is 98% and 46%: luckily, many of the best results belong to the  $J = 2$  grid section.

*Random search.* Another possibility is the random search with some probability distribution on domain  $D$ . The choice of this distribution is an interesting question. The simplest choice is the uniform distribution. Some *a priori* assumptions can be used to adjust it: for example, one would guess that boundary values are less likely to provide good fitting (this turned out to be false). Another guess, also false, was that good solutions are unlikely for low  $\nu$  and zero  $S_0$ . The third guess was that probability is higher in the middle on the domain  $D$  (in the neighbour of the reference solution): this also turned out to be false. Thus, relying on *a priori* guesses is risky from the point of view of performance: in case of a wrong guess random search can be even slower than fixed-order search. Let us estimate the average time needed to collect all good results.



To obtain  $R\%$  of good results we need to check, in average,  $R\%$  of all tasks.

*Heuristic scheduling.* The third way to improve search performance is using heuristic algorithms of task distribution using accumulated information. The problem is the lack of such heuristics and their vulnerability to changes in the model and in the experimental data. Our wrong guesses were based on experience of numerical solving of many similar inverse problems, but in the considered case they did not work fine. Another drawback of such centralized approach is the necessity to control the process by the server which can be overloaded even without this additional work.

*Game-theoretical scheduling.* Further in this paper we study the possibility to apply game-theoretic optimization of the search. Nodes do not just request tasks but also inform the server about distribution of their preferences on  $D$  aiming to make their profit as high as possible. Decisions made by the nodes mutually influence each other, so that the utility functions depend on all the nodes. If the utility functions (in other words, the rules of the game) are chosen well, an equilibrium will be optimal or, at least, better than other search policies.

## 5. Game-theoretical model

Let us consider a computing system with  $m$  computational nodes or players  $C_1, \dots, C_m$  and a set of computational tasks  $B$ . Each node  $C_n$  is characterized by its computational performance  $e_n$ , which is an average number of operations performed in a time unit.

The task set  $B$  is divided into  $n$  non-overlapping blocks  $B = B_1 \cup \dots \cup B_n$  of initial sizes  $N_1, \dots, N_n$ .

To define the priority of a block, we need to estimate the quality of solutions that can be expected to be found there. A threshold  $\epsilon$  is chosen to select good solutions which are those with  $F(s) \leq \epsilon$ , where  $s \in D$  is a grid point.

For each block  $B_i$ , we denote the block quality  $p_i$  as

$$(4) \quad p_i = \sum_{\substack{s \in B_i \\ s \text{ is computed}}} (1 - F(s)/\epsilon)^+, \quad (x)^+ = \max(x, 0), \quad 0 \leq i \leq n$$

If no points from  $B_i$  have been calculated yet,  $p_i = 0$  by its definition.

We define the priority of the block  $B_i$  as

$$\sigma_i = \frac{\frac{1}{n} + p_i}{1 + p_1 + \dots + p_n}, \quad 0 < \sigma_i \leq 1.$$

Assume that all tasks in a fixed block  $B_i$  have the average computational complexity  $\theta_i$ , i.e., a number of operations to process one task.

The nodes make their decisions at time steps  $0, \tau, 2\tau, \dots$ . At each step of the game, each computing node selects exactly one task block. 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  $\tau$  express the amount of useful work performed during this step. This amount depends on the number of finished 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  $B_i$  simultaneously, the more valuable their work is. Let  $n_i$  be the number of the players who have chosen block  $B_i$  at the considered step, and  $\delta(n_i)$  be the congestion coefficient for the block:

$$\delta(n_i) = \frac{m + 1 - n_i}{m} N_i^{\text{av}},$$

where  $N_i^{\text{av}}$  is the number of available (unfinished) tasks in the block.

Then the utility of node  $C_k$  that chooses block  $B_i$  is defined as

$$U_{ki} = \sigma_i \delta(k_i) \frac{e_k}{\theta_i}.$$

Note that

$$(5) \quad U_{kr} = \frac{e_k}{e_i} U_{ir} \quad \forall r, 1 \leq r \leq n$$

Therefore, at each considered time step, we have a singleton congestion game  $G = (C, B, U)$ , where  $C$  is the set of players (computational nodes),  $B$  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  $s = (s_1, \dots, s_m)$ , where the component  $s_i = j$  equals to the block  $B_j$  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 [13] and identical task blocks [14]. 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 [14, 15].

Heterogeneity complicates the model: the utility of each player depends both on its own performance and on task complexity in the chosen block. But due to the form of utility functions, as (5) holds, the game  $G$  has at least one Nash equilibrium in pure strategies [16].

TABLE 2. True fractions of the good and the best results in the blocks. H and L stand for “high” and “low”, respectively.

	HH ..	HL ..	LH ..	LL ..
.. HH	0.047, 0.000	0.025, 0.000	0.070, 0.008	0.038, 0.003
.. HL	0.083, 0.000	0.045, 0.000	0.125, 0.269	0.067, 0.090
.. LH	0.047, 0.000	0.025, 0.000	0.070, 0.014	0.038, 0.005
.. LL	0.083, 0.000	0.045, 0.000	0.125, 0.458	0.067, 0.153

## 6. Application to the inverse problem

Let us divide each parameter span, except  $S_0$ , into two equal parts: of low and of high values. So, for example, the values  $J < 11$  are considered low while  $J > 11$  are high. The span of  $S_0$  is not divided. Then we have 16 blocks  $B_1, \dots, B_{16}$  corresponding to all possible combinations of low and high values for the four parameters.

For the sake of simplicity, let us assume that all nodes have equal unit computational performance  $e_i = 1$  and that all tasks are equally complex with  $\theta_i = 1$ . So, the utility functions  $U_{ki} = U_i$  depend on block only.

There are two algorithms that can be used to find the equilibrium solution or, at least, a schedule that provides better results compared to random search.

The first one is the better- or best-response dynamics algorithm. Such algorithm can be used by the decision-making module of the client software which communicates with the server to request a task from the desired block.

The other approach is to use the fact that utilities of nodes in each block must be equal at an equilibrium (otherwise some nodes would prefer blocks with higher utility). Let us consider this approach in detail.

The players' utilities  $U_i$  are all equal to the same constant  $C/m$ ; then

$$n_i = m + 1 - \frac{C}{\sigma_i N_i^{\text{av}}}.$$

The sum of all  $n_i$  is  $m$ , so

$$C = \frac{(m+1)n - m}{\sum_{j=1}^n \frac{1}{\sigma_j N_j^{\text{av}}}}.$$

Unfortunately, some of  $n_i$  can possibly be negative. This means that some

TABLE 3. Average block qualities  $\bar{p}_i$ .

	HH ..	HL ..	LH ..	LL ..
.. HH	0.85	0.65	0.66	0.68
.. HL	0.77	0.54	0.78	0.61
.. LH	0.74	0.84	0.93	0.85
.. LL	0.85	0.77	0.92	0.70

blocks are too bad, so that high enough utility is impossible there even if  $n_i = 0$ . Then  $n_i = 0$  in these blocks, effective  $n$  is reduced during the step, and  $n_i$  for other blocks are evaluated again. This algorithm produces a non-negative schedule with, possibly, fractional  $n_i$ . They can be used by the server for scheduling purposes. Another possibility is to take rounded values as an approximation to the exact solution. We tested an algorithm for the approximate solution; setting negative  $n_i$  to zero, we normalize positive  $n_i$  by multiplying on

$$\frac{m}{\sum_{i, n_i > 0} n_i}.$$

## 7. Simulation results

We perform simulations basing on the traces of the search performed in [5].

In Table 3 we provide  $p_i$  evaluated after processing all grid nodes and normalized by block sizes, further denoted  $\bar{p}_i$ . It is clear that some blocks are better than the others; also note that there are blocks which are good in general, but have relatively low amount of best results (LHLH). This is an additional challenge for the algorithm, because good block will attract attention, though in the end, it will not give many best results.

We estimate the algorithm efficiency in average, i.e., under the assumption that  $p_i$  obtained using the completed tasks follow the statistical average:  $p_i = \bar{p}_i N_i^{\text{done}}$ , where  $N_i^{\text{done}}$  is the number of computed tasks in block  $i$ .

Using the proposed scheduling algorithm, 90% of best results are collected by the Desktop Grid after 85% of steps are done; 75% need 63%; 50% take less than 42%; finally, 25% are found after 21%. The flow of the best results is shown in Figure 2. In the figure, the proposed algorithm is compared with two other scheduling policies: the random search based on the quality distribution and the “best block first”.

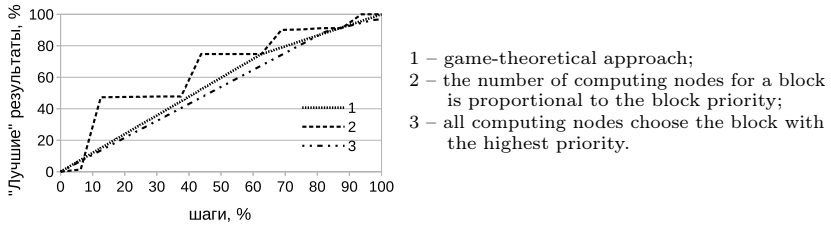


FIGURE 2. Amount of good results produced by three scheduling algorithms

TABLE 4. Stable schedules. Block numbers are left to right, top to bottom along Table 2.

Block	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2nd step	5.0	0.0	0.0	0.0	2.6	0.0	2.9	0.0	1.6	4.7	6.9	5.0	5.0	2.6	6.7	0.2
Stable 1	4.3	0.0	0.0	0.0	4.3	0.0	4.3	0.0	4.3	4.3	4.3	4.3	4.3	4.3	4.3	0.0
Stable 2	0.0	7.8	7.8	7.8	0.0	7.7	0.0	7.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	7.2

The quality measure of the depicted curves is their relative proximity to the upper left corner of the chart. The results illustrate that starting from the very early steps, the game-theoretical algorithm performs better than the one based solely on block priorities. The third considered algorithm depends on available information about true fractions of good results in the blocks. The presented results indicate that it becomes a serious drawback at the early steps of computations, when there is not enough gathered information.

Simulations showed that the initial uniform distribution of computing nodes over task blocks quickly converges to a stable configuration. Some blocks are left uninvestigated while more promising ones are not empty. The best blocks are studied more, so that soon the schedule becomes uniform on better nodes and zero on worse ones. When mostly studied blocks have no more tasks left, the schedule changes and converges to another one, investigating less promising blocks. The gradient jump in Figure 2 marks this point: after it, the intensity of good results production decreases.

Both stable schedules are shown in Table 4. The table cells contain fractional numbers of the computing nodes that choose the corresponding block.

The proposed scheduling algorithm is flexible: it is able to adjust itself “on the fly” as long as new information arrives, computing nodes leave or join the Desktop Grid, or the blocks configuration changes.

There are a few possibilities to further improve the described approach to scan the parameter space. First, we plan to increase the number of distinguishable blocks. Then the high quality of one block will make its neighbour blocks more attractive. Also, configuration of the blocks can be variable; promising blocks can be further divided into smaller ones, and the worst blocks can be grouped.

The choice of the utility function is an important question. In the studied example, worse blocks were hardly studied at all up until better blocks would be finished. It seems safer to force at least a few nodes to study bad blocks in order to accumulate and update statistics. However, a few steps can be done under the uniform schedule to gather information about block quality.



Finally, we plan to test the algorithm for more complex search problems: denser grids, continuous blocks, problems with more parameters, heterogeneous Desktop Grids of larger scale. If computations require weeks or even months, the fast production of good results becomes extremely important.










## Conclusion

We consider a parameter identification problem and solve it by scanning over the parameter space in parallel. To study methods of effective search, we focus on a simple, yet useful, example: a relatively small grid in the parameter space was scanned by a small-size Enterprise Desktop Grid.

We propose a game-theoretic algorithm of task scheduling over subsets of the parameter space called blocks and study its performance by comparing with two heuristics. Simulations show that the algorithm allows to gain performance increase in terms of the good results discovery rate.

## References

- [1] R. C. Aster, B. Borchers, C. H. Thurber. *Parameter estimation and inverse problems*, Elsevier, 2005, 301 pp. <sup>↑</sup><sub>54</sub>
- [2] A. N. Tikhonov, V. Y. Arsenin. *Solutions of ill-posed problems*, Wiley-Intersci. Publ., New York, 1977. <sup>↑</sup><sub>54</sub>
- [3] I. Foster, C. Kesselman. *The grid: Blueprint for a new computing infrastructure*, Morgan Kaufmann Inc., San Francisco, CA, USA, 1999. <sup>↑</sup><sub>54</sub>
- [4] I. Foster, C. Kesselman, S. Tuecke. “The anatomy of the grid: Enabling scalable virtual organizations”, *International J. Supercomputer Applications*, **15:3** (2001), pp. 200–222.  <sup>↑</sup><sub>54</sub>
- [5] I. A. Chernov, E. E. Ivashko, N. N. Nikitina, I. E. Gabis. “Numerical identification of the dehydriding model in a BOINC-based grid system”, *Computer Research and Modeling*, **5:1** (2013), pp. 37–45 (in Russian).  <sup>↑</sup><sub>55,58,64</sub>

- [6] I. A. Chernov, I. E. Gabis. “Mathematical modelling of hydride formation”, *Mathematical Modeling, Clustering Algorithms and Applications*, Mathematics Research Developments, ed. C. L. Wilson, Nova Science Publishers, 2011, pp. 203–246. <sup>↑</sup><sub>56</sub>
- [7] I. E. Gabis, I. A. Chernov, *Kinetics of decomposition of binary metal hydrides*, Chemistry Research and Applications, Nova Science Publishers, 2017, 137 pp. <sup>↑</sup><sub>56</sub>
- [8] I. E. Gabis, A. P. Voyt, I. A. Chernov, V. G. Kuznetsov, A. P. Baraban, D. I. Elets, M. A. Dobrotvorsky. “Ultraviolet activation of thermal decomposition of  $\alpha$ -alane”, *International Journal of Hydrogen Energy*, **37**:19 (2012), pp. 14405–14412.  <sup>↑</sup><sub>56</sub>
- [9] S. V. Manicheva, I. A. Chernov. “Mathematical model of hydride phase change in a symmetrical powder particle”, *Computer Research and Modeling*, **4**:3 (2012), pp. 569–584 (in Russian).  <sup>↑</sup>
- [10] D. P. Anderson. “BOINC: A system for public-resource computing and storage”, *GRID '04 Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, IEEE Computer Society, 2004, pp. 4–10.  <sup>↑</sup><sub>58</sub>
- [11] E. Ivashko. “Enterprise desktop grids”, CEUR Workshop Proceedings, vol. **1502**, Proceedings of the Second International Conference BOINC-based High Performance Computing: Fundamental Research and Development (BOINC:FAST 2015), 2015, pp. 16–21.  <sup>↑</sup><sub>58</sub>
- [12] M. A. Khan. “A survey of computation offloading strategies for performance improvement of applications running on mobile devices”, *Journal of Network and Computer Applications*, **56** (2015), pp. 28–40.  <sup>↑</sup><sub>58</sub>
- [13] R. W. Rosenthal. “A class of games possessing pure-strategy Nash equilibria”, *International Journal of Game Theory*, **2**:1 (1973), pp. 65–67.  <sup>↑</sup><sub>62</sub>
- [14] I. Milchtaich. “Congestion games with player-specific payoff functions”, *Games and Economic Behavior*, **13**:1 (1996), pp. 111–124.  <sup>↑</sup><sub>62</sub>
- [15] S. Jeong, R. McGrew, E. Nudelman, Y. Shoham, Q. Sun. “Fast and compact: A simple class of congestion games”, *AAAI'05 Proceedings of the 20th national conference on Artificial intelligence*. V. 2, 2005, pp. 489–494.  <sup>↑</sup><sub>62</sub>
- [16] M. Gairing, M. Klimm. “Congestion games with player-specific costs revisited”, *Algorithmic Game Theory*, SAGT 2013, Lecture Notes in Computer Science, vol. **8146**, ed. B. Vöcking, Springer, Berlin–Heidelberg, 2013, pp. 98–109.  <sup>↑</sup><sub>62</sub>


Received	29.12.2017
Revised	23.10.2018
Published	01.11.2018


Recommended by

Programm Committee of the  
National Supercomputing Forum NSCF-2017

*Sample citation of this publication:*

Ilya Chernov, Natalia Nikitina. “Effective scanning of parameter space in a Desktop Grid for identification of a hydride decomposition model”. *Program Systems: Theory and Applications*, 2018, **9**:4(39), pp. 53–68.

 10.25209/2079-3316-2018-9-4-53-68

 [http://psta.psir.as.ru/read/psta2018\\_4\\_53-68.pdf](http://psta.psir.as.ru/read/psta2018_4_53-68.pdf)

The same article in Russian:  10.25209/2079-3316-2018-9-4-35-52

*About the authors:***Ilya Aleksandrovich Chernov**

PhD in Physics and Mathematics, senior research fellow of IAMR KarRC RAS. Associate professor of the Chair of Mathematical Analysis PetrSU. Expert in the field of mathematical modeling of physical processes and boundary problems of mathematical physics. Area of scientific interests: numerical analysis, modeling the metals hydrides decay, numerical modeling of high-scale sea dynamics.



0000-0001-7479-9079

e-mail: [chernov@krc.karelia.ru](mailto:chernov@krc.karelia.ru)

**Natalia Nikolaevna Nikitina**

PhD in Technics, junior research fellow of IAMR KarRC RAS. Area of scientific interests: high-performance and distributed computing, task scheduling, Desktop Grid, applications of mathematical game theory.



0000-0002-0538-2939

e-mail: [nikitina@krc.karelia.ru](mailto:nikitina@krc.karelia.ru)

Эта же статья по-русски:



10.25209/2079-3316-2018-9-4-35-52