ISSN 0146-4116, Automatic Control and Computer Sciences, 2019, Vol. 53, No. 8, pp. 1038–1044. © Allerton Press, Inc., 2019.
Russian Text © The Author(s), 2019, published in Problemy Informatsionnoi Bezopasnosti, Komp'yuternye Sistemy.

Model and Method for Optimizing Computational Processes in Parallel Computing Systems

V. G. Anisimov[a], *, P. D. Zegzhda[a], **, E. G. Anisimov[b], T. N. Saurenko[b], and V. V. Kasatkin[c]

[a] *Peter the Great Polytechnic University St. Petersburg, St. Petersburg, 195251 Russia*

[b] *RUDN University, Moscow, 117198 Russia*

[c] *St. Petersburg Institute for Computer Science and Automation, Russian Academy of Sciences, St. Petersburg, 199178 Russia*

**e-mail: an-33@yandex.ru*

***e-mail: zeg@ibks.spbstu.ru*

Received November 12, 2018; revised November 22, 2018; accepted December 1, 2018

Abstract—The paper proposes a model and a method for optimizing computational processes in parallel computing systems. The model is constructed by the decomposition of computer programs for solving problems into relatively independent elements (blocks) and presenting the information dependence of the blocks in the form of corresponding directed graphs. Here, the computational process organization consists in the dynamic allocation of resources of a computing system for the implementation of operations of each block. As an efficiency indicator of this process, we take the time of implementing operations of all blocks of considered programs (the solution time of related problems). The goal of the optimization is to reduce this time as much as possible. To form the optimal resource allocation, we propose the method based on the branch-and-bound procedure.

Keywords: computer, parallel computing-system, computational process organization, optimization, model, method

DOI: 10.3103/S0146411619080054

INTRODUCTION

A characteristic feature of the modern stage of computer development is the advent and widespread use of large-scale computers and supercomputers. Their appearance is due to the need to rapidly resolve complex information and calculation tasks in various spheres of action [1–18]. Solving such tasks is associated with the necessity to handle enormous volumes of information nearly impossible by von Neumann computers. A way out of this situation at present and in the foreseeable future is ensured by a parallelization of corresponding algorithms and using parallel computing systems (large-scale computers and supercomputers). However, the efficiency of using such systems for solving a given problem, largely depends on the compatibility of the structure of a computer program that implements an algorithm for solving the problem and organizes its execution with features of the computer architecture. For example, in programs targeted at a vector-pipeline computer, it is necessary to vectorize internal cycles, while in programs targeted at a parallel computer with a shared memory or a computer cluster with a distributed memory, we should parallelize pieces of a program and efficiently organize their execution. Here, the parallelization as well as the organization of programs running on a specific computer depend on the specifics of the task and represent problems of their developers [19, 20]. The aim of this paper is to propose tools for solving the problem of the optimal computational process organization (CPO) in parallel computers with a shared or distributed memory. The development of these tools consists in constructing a model and a method that ensure the problem-time optimal CPO.

DESCRIPTION OF THE MODEL

An application problem solved with the use of computer equipment is presented as a corresponding computer program. This program represents a given sequence of operations that convert input data into output data. An operation cannot start before the end of all operations that are suppliers of input data required for it. Hence, the computer program sets the partial order of execution of included operations

[19]. For each pair of operations this order specifies an operation that must be executed earlier or notes that the operations are independent and can be executed in any order. If the program does not contain independent operations, then the order is linear and only one variant for the program's implementation is available. In the case of independent operations, a fairly wide range of implementation variants can be found and the problem of choosing the best of them arises. For benefit of the choice, the criterion of optimality is specified and the computer program is decomposed into relatively independent blocks. Each block contains one or more related operations. The information interrelation of the blocks is presented by the directed graph

$$G = \{(i, j)\}, \quad i, j = 0,1,\ldots,m, \quad i < j, \tag{1}$$

where i and j are node numbers of this graph, m is the number of allotted relatively independent blocks of the program, and $(m+1)$ is the number of the nodes of the graph.

Each computational procedure in this graph is placed in correspondence with an arc (i, j), which connects the ith and jth nodes. The node $i = 0$ represents the event, which consists in starting the computer program. The nodes $i = 1, 2,\ldots, m$ represent the events, which consist in the completion of all computational procedures corresponding to arcs incoming to each of them. Here, the mth node represents the event, which consists in the completion of executing the computer program (the completion of problem solving).

The sequence of computational procedures obeys the following condition: a procedure corresponding to an arc coming from any node cannot be started until the completion of all procedures corresponding to the arcs incoming to this node is found.

The possibilities of a computing system for operations that compose each of the arcs are specified by the vector

$$T = \|t_k(i, j)\|, \quad (i, j) \in G(i, j), \quad k \in R, \tag{2}$$

where $t_k(i, j)$ is the time required for the kth element of the computing system to execute the procedure corresponding to the (i, j)th arc; k is the identifier of an element of the computing system; and R is the set of elements of the computing system.

If the kth element of the computing system cannot execute operations of the procedure corresponding to the (i, j)th arc, then $t_k(i, j) = \infty$.

Assume that with the procedure involving the execution of operations corresponding to the (i, j)th arc of a certain set $W(i, j)$ of elements of the computing system, their execution time $t(i, j)$ is determined by a certain function

$$t(i, j) = t[W(i, j)], \quad (i, j) \in G(i, j). \tag{3}$$

Its constructive presentation depends on the specific operations of the procedure.

The CPO is defined by the set

$$P = \{A_P(i, j), W_P(i, j)\}, (i, j) \in G(i, j), W_P(i, j)) \subset R, \tag{4}$$

where $A_P(i, j)$ is a moment corresponding to the start of the nth procedure of the program in implementing the P variant of the CPO and $W_P(i, j)$ is a set of elements of the computing system that are involved in executing the (i, j)th procedure of the program when implementing the P variant of the CPO.

It is assumed that an interruption of each started procedure is not allowed and the composition $W_P(i, j)$ of involved elements is not changed during the procedure's execution.

The set of all paths of directed graph (1) that connect its initial and terminal nodes we denote by G_L. The execution time $T(P)$ of the computer program is equal to the maximum time $T_L(P)$ of the travel $L \in G_L$ from the initial node $i = 0$ of graph (1) to its terminal node $i = m$ in implementing the P variant of the CPO.

With allowance for the accepted notation and constraints, a model for the choice of an optimum variant of the CPO can be formally presented as the following mathematical programming problem: specify the variant

$$P^* = P\left\{A_P^*(i, j), W_P^*(i, j)\right\}, \quad (i, j) \in G(i, j), \quad W_P^*(i, j) \subset R \tag{5}$$

(of the organization of a computational process implementing the execution of the computer program described by directed graph (1)) satisfying the condition

$$T_L(P^*) = \min_P \max_{L \in G_L} T_L(P) \tag{6}$$

under the constraints

$$A_P(i,j) \geq \max_{(z,j)\in G}\{A_P(z,j) + t_P(z,j)\}, \quad (i,j) \in G(i,j), \quad (z,j) \in G(i,j); \tag{7}$$

$$\sum_{(i,j)\in F_P(t)} \sum_{k\in W_P(i,j)} d_k(i,j) \leq K, \tag{8}$$

where $F_P(t)$ is a set of individual processes executed at the moment t for the P variant of the CPO;

$$d_k(i,j) = \begin{cases} 1, & \text{if} \quad k \in W_P(i,j) \\ 0, & \text{if} \quad k \notin W_P(i,j) \end{cases}; \tag{9}$$

and K is the power (the number of elements) of the set R.

Condition (6) in model (5)–(9) formally reflects the desire to minimize the execution time of the computer program. Condition (7) establishes that computational procedures corresponding to an arc coming from any node of graph (1) can start only after the completion of all procedures represented by the arcs incoming to this node. Condition (8) establishes that the number of simultaneously used elements of the computing system cannot exceed their total number.

In general, a computational process (organized on model (5)–(9)) that implements the execution of the computer program whose structure is described by directed graph (1), provides the minimization of the solution time of the corresponding applied problem.

METHOD FOR OPTIMIZING THE COMPUTATIONAL PROCESS

Model (5)–(9) represents a nonlinear integer programming problem. It belongs to the class of non-polynomially complex tasks. A unified optimization algorithm for problems of this type is not found. Almost all of them require a special algorithm that takes into account its specificity [19–25]. In this regard, consider an algorithm that takes into account the specificity of model (5)–(9). For the basis of the algorithm we take the branch-and-bound procedure [26, 27]. It contains a finite number of steps and rests on the following constructions [28, 29]:

(a) Representing the set $V = \{g\}$ of fragments g (feasible for constraints (7)–(9)) of a P plan of the CPO, as a tree of subsets (branching).

(b) Computing the low bound of goal function (5) for these fragments (branches).

(c) Finding acceptable variants of the P plan of the CPO.

(d) Checking these variants for optimality.

Branching in the proposed algorithm is implemented on the basis of a dichotomous scheme. In its implementation, every v_gth node of the gth branch of the tree of variants represents an element of the plan. Here, if the (i,j)th computational procedure corresponding to this element starts at the $A_g(i,j)$th moment for the $W_g(i,j)$th variant of bringing elements of the computing system to the execution of the procedure, then

$$v_g = \{A_g(i,j), W_g(i,j)\}. \tag{10}$$

If the (i,j)th computational procedure does not do this, then

$$v_g = \varnothing. \tag{11}$$

Here, $A_g(i,j), (i,j) \in g$ (starting points for corresponding procedures) for each branch $g \in V$ must be chosen from the ascending sequence corresponding to this branch

$$\beta_g = \{t_g^n\}, \quad n = 1,2,\ldots. \tag{12}$$

In this case, $t_g^1 = 0$ and the subsequent moments $t_g^n, n = 2,3,\ldots$ are determined recurrently by the formula

$$t_g^n = \min_{(i,j)\in F_g(t_g^{n-1})} \{A_g(i,j) + t_g(i,j)\}, \quad n = 2,3,\ldots, \tag{13}$$

where $F_g\left(t_g^{n-1}\right)$ is a set of procedures (i,j) previously included in the gth branch and uncompleted to the point in time t_g^{i-1}; i.e.,

$$F_g\left(t_g^{n-1}\right) = \left\{(i,j) \mid (i,j) \in g, A_g(i,j) \leq t_g^{n-1} \leq A_g(i,j) + t_g(i,j)\right\}. \tag{14}$$

Here, $t_g(i,j)$ in (14) is determined by relation (3) for $W(i,j) = W_g(i,j)], (i,j) \in G(i,j)$.

Hence, β_g represents a sequence of points in time at which the execution of computational procedures included in the considered branch of the tree of variants ends and involved elements of the computing system are released. Here, the condition $t_g^1 = 0$ reflects the fact that all variants of implementing the computer program corresponding to graph (1) start at the moment $t = 0$.

The timing of the beginning of the processes in breach of the specified sequences makes it impossible to reduce the total execution time of the program. In fact, for any plan P (of the CPO) containing the gth fragment, the early start of any procedure $(i, j) \in g$ belongs to sequence (12) by definition. Consequently, later starts of computing procedures lying on paths critical for the P plan also belong to this sequence. For procedures that do not belong to critical paths, variation of the start times within corresponding reserves of time is possible. However, the limits of these reserves also belong to sequence (12) and the variation within the limits does not change the total execution time of a computer program. Consequently, the start and end times of procedures of (optimal for criterion (6)) plan (5) of the CPO must belong to the sequence β_g corresponding to this plan.

To implement the dichotomic branching scheme, we introduce the set of all possible variants of the assignment of computer system's elements for implementing each (i,j)th procedure (arc) of graph (1)

$$U = \{u_q(i,j)\}, (i,j) \in G, \quad q = 1,2,\ldots, \tag{15}$$

where $u_q(i,j) = \begin{cases} 1, & \text{if for executing the } (i,j)\text{th procedure, the system involves a set} \\ & W^q(i,j) \text{ of elements of the computer network;} \\ 0, & \text{if for executing the } (i,j)\text{th procedure, the system does not involve a set} \\ & W^q(i,j) \text{ of elements of the computer network.} \end{cases}$

Then, the position number q of the element $u_q(i,j) = 1$ of the set U characterizes an executed computational procedure and a variant of bringing elements of the computing system to the procedure's execution.

With allowance for (15), a branching process for the benefit of forming optimal plan (5) of the CPO, consists in choosing (for each ordinary moment $t_g^i \in \beta_g$) admissible variables $u_q(i,j) \in U$ and specifying their values; i.e., relation (10) takes the form $\nu_g = \{t_g^n, u_q(i,j) = 1\}$ if the procedure $(i,n)\, j \in G$ corresponding to the variable $u_q(i,j)$ starts at the moment $A_g(i,j) = t_g^n$ for the $W_g(i,j) = W^q(i,j)$th variant of the assignment of elements of the computer network for the procedure's execution, or $\nu_g = \{t_g^n, j_q(i,j) = 0\}$ if this process for the considered variant of the resource assignment does not start at the moment $A_g(i,j) = t_g^n$.

The set $U_g^n \subset U$ of variables $u_q(i,j) \in U$, which can be included in the gth fragment of a plan of the CPO at the moment t_g^n, contains values of $u_q(i,j) \in U$ corresponding to procedures $(i,j) \in G$ previously not included in the gth branch; for them

$$A(l,i) + t(l,i) \le t_g^n, \quad (l,i) \in G; \tag{16}$$

$$W^q(i,j) \subseteq R_g^n. \tag{17}$$

Here, R_g^n is a set of free elements of the computing system for the gth fragment of the plan of the CPO at the moment t_g^n and $U_g(t_g^{n-1})$ is a set of variables $u_q(i,j)$ included in the gth branch of the tree of variants of a plan of the CPO to the moment t_g^{n-1}.

Condition (16) allocates computational procedures for which all procedures that represent suppliers of input data required for them are executed at the moment t_g^n. Condition (17) selects those of them for which there are free elements of the computer network.

As an estimate Q_g of the lower bound of goal function (6) for each gth fragment of the production plan, we can take the maximum time T_L^g of travel from the initial node $i = 0$ of graph (1) to its terminal node; this time is determined without regard to the constraints on allotting elements of the computer network for processes not included in g, i.e., belonging to the set G^*, which supplements g to the set of procedures

of graph (1). Here, if at the subsequent branching step (corresponding to the moment t_g^n) for the procedure $(i, j) \in G$ it is assigned $u_q(i, j) = 1$, then for determining $Q_g\left(u_q(i, j) = 1\right)$ we have the following items:

(a) The procedures $(l,i) \in g$ previously included in the gth fragment of the plan (i.e., procedures for which $A_g(l,i) < t_g^n$), start at corresponding moments $A_g(l,i)$ and end at the moments $A_g(l,i) + t_g(l,i)$ (here, $t_g(l,i)$ are determined by relation (3)).

(b) For the procedure $(i, j) \in G^* \subset G$ corresponding to the variable $u_q(i, j) = 1$ and, hence, included at a considered step in the gth branch of the tree of variants, the start time is $A_g(i, j) = t_g^n$ and the completion time $t_g^n + t_g(i, j)$ ($t_g(i, j)$is determined by relation (3)).

(c) For the procedures $(e,h) \in G^* \subset G$ (corresponding to the variables $u_r \in U, r \neq q$), which by constraint (16) at the moment t_g^n cannot be executed at the same time with (i, j), the start time is t_g^{n+1} and the completion time is determined by the relation $t_g^{n+1} + t^*(e,h)$, where

$$t^*(e,h) = \min_{W(e,h)} t[W(e,h)], \quad (e,h) \in G^* \subset G, \quad W(e,h) \subset R. \tag{18}$$

Here, if at the considered branching step it is assigned $u_q(i, j) = 0$, then for determining $Q_g\left(u_q = 0\right)$ we additionally have the following items:

(a) Each procedure $(l,i) \in G^* \subset G$ previously included in the gth fragment of the plan (i.e., a procedure for which $A_g(l,i) < t_g^n$), starts at the corresponding moment $A_g(l,i)$ and ends at the moment $A_g(l,i) + t(l,i)$ ($t(l,i)$ is determined by relation (3)).

(b) The procedure (i, j) corresponding to the variable $u_q(i, j) = 0$, starts at t_g^{n+1} and ends at $t_g^{n+1} + t^*(l,i)$.

(c) The remaining the procedures $(e,h) \in G^* \subset G$ corresponding to the variables $u_r \in U_g^n, u \neq q$, start at t_g^n and end $t_g^n + t^*(e,h)$.

An important element of the method for solving problem (5)–(9) is a way of choosing the subsequent procedure and a variant of assigning elements of the computer network for executing the procedure: this element significantly affects the convergence of the method. Formally, this way consists in choosing the variables $u_q \in U_g^n$ for inclusion in the gth branch at t_g^n. In the proposed method, at the subsequent branching step, the variable $u_q \in U_g^n$ is chosen; for this variable,

$$t^*(i, j) = \min_{W(i,j)} t[W(i, j)] \xrightarrow[(i,j) \in G^*]{} \max, \; W(i, j) \subset R. \tag{19}$$

If several variables of this kind are found, then the choice of them is carried out in accordance with the following rule of preference:

$$\min i \rightarrow \min j; \tag{20}$$

i.e., the variable corresponding to the procedure with the smallest number i is first included in a branch of the tree of variants. If, here, several variables of this kind are found, then the variable corresponding to the procedure with the smallest number j is chosen from them.

The tree traversal is organized in accordance with the "go right" rule. This allows us to store in a computer memory when solving the problem only the current fragment of the plan, the smallest value of previously obtained values of the goal function, and a feasible plan (of the CPO) corresponding to this value.

This rule, combined with the considered method of choosing computational procedures and elements of the computing system involved in their implementation, represents an approximate algorithm for solving problem (5)–(9); this algorithm allows us to obtain the first feasible solution in a finite number of steps equal to the number N of arcs of the graph G.

Each gth branch ends if it includes all N procedures (arcs of the graph G), i.e., the feasible P plan of the CPO is obtained, or if

$$Q_g \geq T^0(1 - \mu), \quad 0 \leq \mu \leq 1, \tag{21}$$

where T^0 is the minimum value of the goal function for previously obtained feasible plans (a record) and μ is the assigned permissible relative deviation of the goal function from the optimal value (the accuracy of optimization).

The fulfilment of condition (21) means that it is impossible to improve a previously obtained plan on a considered branch more than by $100\mu\%$ and its continuation within the given accuracy of optimization is meaningless. The procedure for finding a solution ends if condition (21) holds for all remaining branches.

With the assigned way for traversing the tree of variants, such a situation is matched by the second return to the root node. Here, the last record represents the sought value of goal function (10) and feasible plan (5) corresponding to this value and represents the optimal P^* plan of the CPO for the program whose structure is presented by graph (1).

CONCLUSIONS

Using the this model and method ensures the formation of optimal plans of the CPO in computing systems with a shared or distributed memory. In the context of the model we propose the procedure of forming a plan of the CPO; the procedure provides the analysis of all possible variants for such an organization and excludes duplicates when viewing them. Here, for its implementation it is sufficient to store in the computer memory only the current fragment of the plan for organizing the process, the smallest value of previously obtained values of goal function (5), and a corresponding feasible plan. This significantly reduces the cost of computer resources in the formation of the plan in question. The proposed rule for traversing a tree of variants, combined with the method of choosing, at every branching step, the computational procedures and elements of the computing system involved in their implementation, represents an approximate algorithm for solving problem (5)–(9); this algorithm allows us to obtain the first feasible plan of the CPO for the finite number of steps equal to the number N of arcs of graph (1).

In general, the proposed model and method allow optimizing a computational process in parallel computing systems.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. Il'in, I.V., et al., *Matematicheskie metody i instrumental'nye sredstva otsenivaniya effektivnosti investitsii v innovatsionnye proekty* (Mathematical Methods and Tools for Assessing the Effectiveness of Investments in Innovative Projects), St. Petersburg, 2018.
2. Anisimov, V.G., Anisimov, E.G., Zegzhda, P.D., Saurenko, T.N., and Prisyazhnyuk, S.P., Indices of the effectiveness of information protection in an information interaction system for controlling complex distributed organizational objects, *Autom. Control Comput. Sci.,* 2017, vol. 51, no. 8, pp. 824–828.
3. Gasyuk, D.P., Sosyura, O.V., et al., *Osnovy teorii effektivnosti boevykh deistvii raketnykh voisk i artillerii* (Fundamentals of the Theory of the Effectiveness of Combat Missile Forces and Artillery), Moscow: Minist. Oborony Ross. Fed., 2003.
4. Saurenko, T.N., Tebekin, A.V., et al., *Ekonomicheskii i tamozhennyi risk-menedzhment* (Economic and Customs Risk Management), Moscow: Ross. Tamozhennaya Akad., 2015.
5. Saurenko, T.N., Tebekin, A.V., Tebekin, P.A., et al., *Teoreticheskie osnovy upravleniya innovatsiyami* (Theoretical Foundations of Innovation Management), St. Petersburg, 2016.
6. Anisimov, V.G., Anisimov, E.G., and Bogoeva, E.M., Formalization of the procedure of the risk-based approach in the performance of control functions by state bodies, *Vestn. Ross. Tamozhennoi Akad.,* 2014, no. 4, pp. 96–102.
7. Kezhaev, V.A., Svertilov, N.I., et al., *Metody i modeli optimizatsii v upravlenii razvitiem slozhnykh tekhnicheskikh sistem* (Methods and Optimization Models in Managing the Development of Complex Technical Systems), St. Petersburg, 2004.
8. Alekseev, O.G., et al., *Modeli raspredeleniya sredstv porazheniya v dinamike boya* (Models of the Distribution of Weapons in the Dynamics of Battle), Leningrad: Minist. Oborony SSSR, 1989.
9. Anisimov, E.G., Anisimov, V.G., and Sonkin, M.A., Mathematical simulation of adaptive allocation of discrete resources, *Proceedings of the 2016 Conference on Information Technologies in Science, Management, Social Sphere and Medicine (ITSMSSM 2016),* 2016, pp. 282–285.

10. Gar'kushev, A.Yu. Sazykin, A.M., et al., Methodological provisions of mathematical modeling of problems of adaptive distribution of discrete resources in the management of troops and weapons in real time, *Izv. Ross. Akad. Raketnykh Artilleriiskikh Nauk,* 2016, no. 1, pp. 32–37.
11. Anisimov, V.G., Anisimov, E.G., Saurenko, T.N., and Sonkin, M.A., The model and the planning method of volume and variety assessment of innovative products in an industrial enterprise, *J. Phys.: Conf. Ser.,* 2017, vol. 803, no. 1, p. 012006. https://doi.org/10.1088/1742-6596/803/1/012006
12. Gar'kushev, A.Yu. Selivanov, A.A., et al., Performance indicators of interagency information interaction in the management of state defense, *Vopr. Oboronnoi Tekh., Ser. 16: Tekh. Sredstva Protivodeistviya Terrorizmu,* 2016, nos. 7–8, pp. 12–16.
13. Saurenko, T.N., Gapov, M.R., Anisimov, V.G., Anisimov, E.G., and Mekala, S.K., Formalization of planning procedure—production process of the complex industrial patterns of vertical integration, *Ekonomicheskie strategii EAES: Problemy i innovatsii. Sbornik materialov Vserossiiskoi nauchno-prakticheskoi konferentsii* (EAEU Economic Strategies: Problems and Innovations. Proc. All-Russian Scientific-Practical Conference), 2018, pp. 154–161.
14. Anisimov, V.G., Zegzhda, P.D., Anisimov, E.G., and Bazhin, D.A., A risk-oriented approach to the control arrangement of security protection subsystems of information systems, *Autom. Control Comput. Sci.,* 2016, vol. 50, no. 8, pp. 717–721.
15. Sazykin, A.M., et al., The theoretical basis for creating decision support systems in the interests of integrated transport security, *Izv. Ross. Akad. Raketnykh Artilleriiskikh Nauk,* 2015, no. 3, pp. 10–15.
16. Poltavtseva, M.A., Evolution of data management systems and their security, *Proceedings—2019 International Conference on Engineering Technologies and Computer Science: Innovation and Application,* 2019, article no. 8711971, pp. 25–29. https://doi.org/10.1109/EnT.2019.00010
17. Poltavtseva, M.A., Zegzhda, P.D., and Pankov, I.D., The hierarchial data aggregation method in backbone traffic streaming analyzing to ensure digital systems information security, *2018 Eleventh International Conference "Management of Large-Scale System Development" (MLSD),* Moscow, 2018, pp. 1–5. https://doi.org/10.1109/MLSD.2018.8551916
18. Lavrova, D., Poltavtseva, M., and Shtyrkina, A., Security analysis of cyber-physical systems network infrastructure, *Proceedings 2018 IEEE Industrial Cyber-Physical Systems,* 2018, pp. 818–823. https://doi.org/10.1109/ICPHYS.2018.8390812
19. Poltavtseva, M.A., Lavrova, D.S., and Pechenkin, A.I., Planning of aggregation and normalization of data from the Internet of Things for processing on a multiprocessor cluster, *Autom. Control Comput. Sci.,* 2016, vol. 50, no. 8, pp. 703–711. https://doi.org/10.3103/S0146411616080162
20. Kalinin, M., Lavrova, D., and Pechenkin, A., High performance traffic processing in virtualized framework, *C. R. Acad. Bulg. Sci.,* 2015, vol. 68, no. 7, pp. 909–916.
21. Anisimov, V., Chernysh, A., and Anisimov, E., Model and algorithm for substantiating solutions for organization of high-rise construction project, *E3S Web of Conferences Ser. High-Rise Construction 2017, HRC* 2017, 2018, p. 03003.
22. Anisimov, V., Anisimov, E., and Sonkin, M., A resource-and-time method to optimize the performance of several interrelated operations, *Int. J. Appl. Eng. Res.,* 2015, no. 10, pp. 38127–38132.
23. Alekseyev, A.O., Alekseyev, O.G., Anisimov, V.G., Anisimov, Ye.G., and Yachkula, N.I., Application of Markov chains in estimating the computational complexity of the simplex method, *Sov. J. Comput. Syst. Sci.,* 1988, no. 5, pp. 130–134.
24. Chvarkov, S.V., et al., *Matematicheskie metody i modeli v voenno-nauchnykh issledovaniyakh* (Mathematical Methods and Models in Military Scientific Research), Moscow: Akad. Gen. Shtaba Vooruzhennykh Sil Ross. Fed., 2017, part 1.
25. Chvarkov, S.V., et al., *Matematicheskie metody i modeli v voenno-nauchnykh issledovaniyakh* (Mathematical Methods and Models in Military Scientific Research), Moscow: Akad. Gen. Shtaba Vooruzhennykh Sil Ross. Fed., 2017, part 2.
26. Anisimov, V.G. and Anisimov, Ye.G., A branch-and-bound algorithm for one class of scheduling problem, *Comput. Math. Math. Phys.,* 1992, vol. 32, no. 12, pp. 1827–1832.
27. Anisimov, V.G. and Anisimov, E.G., A method of solving one class of integer programming problems, *Comput. Math. Math. Phys.,* 1989, vol. 29, no. 5, pp. 238–241.
28. Anisimov, V.G. and Anisimov, E.G., The algorithm for the optimal distribution of discrete heterogeneous resources on the net, *Comput. Math. Math. Phys.,* 1997, vol. 37, no. 1, pp. 54–60.
29. Anisimov, V.G. and Anisimov, E.G., Modification of the method for solving a class of integer programming problems, *Comput. Math. Math. Phys.,* 1997, vol. 37, no. 2, pp. 179–183.

Translated by L. Kartvelishvili