Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EVALUATION OF COMPLEXITY OF THE LARGE-BLOCK CLOUD COMPUTING USING ARITHMETIC WITH ENHANCED ACCURACY

[1]VLADIMIR EFIMOVICH PODOLSKIY, [2]SERGEY STEPANOVICH TOLSTYKH
[3]ANTON MIKHAILOVICH BABICHEV, [4]SVETLANA GERMANOVNA TOLSTYKH

[1]Tambov Regional Center of New Information Technologies
[2,3,4]Tambov State Technical University
Sovetskaya Str., 106, Tambov, 392000 Russia

E-mail:[1]director@director.tixmcnit.tambov.su, [2]inf@tstu.ru, [3]zhelkzhelk@yandex.ru,
[4]svetlanatolstyh@mail.ru

ABSTRACT

In the article, the methodological questions are considered of the complexity evaluation of large-block cloud computing with enhanced accuracy. The objective of our work is solving in the cloud of the mathematical simulation problems with special requirements for accuracy. In particular, we consider obtaining a precise and trustworthy solution for the problems with complex connections between sub-problems in the form of large blocks and the computation time considerably exceeding the time of information transmission between them. A methodology is proposed of the complexity evaluation for such problems used for creation of the optimal productivity computing systems, operating in a cloudy environment.

Keywords: *Large-Block Parallel Computing, Cloud Computing, Precision Computation, Precise And Trustworthy Solution Of Problems, Floating Point Arithmetic With Enhanced Accuracy.*

1. INTRODUCTION

The objective of the work is the cloud-oriented solving of a broad class of problems based on mathematical models with a complex topology. In addition, this class is characterized by a large volume of floating-point calculations with stricter requirements for the result's reliability. Among such problems there are, for example, the problem of finding a partial system of combinations of the works on the water body recreation in an industrial region, the problem of designing a ventilation system in the enclosed bounded spaces with a complex configuration and many others. The problems of this kind require a responsible solution, because the obtained results may influence the volume of capital investments, life safety, or the rated quality of products. Similar problems were considered in the 80s in the works of G.M. Ostrovsky [1-3] devoted to decomposition of complex chemical-engineering schemes. In reality, the gain in computation time on the uniprocessor computers was achieved by finding an optimal iterated set (OIS) and reducing the dimension of the overall system of nonlinear equations; it was possible to find the cuts of the digraph of technological system such that the total number of variables used for iterative calculation of the mathematical model was minimal (for example, a model of chemical plant with the number of process vessels about 100). The problem of optimal decomposition was solved by the branch-and-bound method, by the ordered exhaustive search of the digraph arcs, intended to be cut, on the matrix of contours, and by inclusion of the variables corresponding to the arc to an OIS of minimal cardinality. After finding the OIS, global iterations were performed by the Newton-Raphson method or its modifications: instead of a huge system of nonlinear equations (SNE), it was possible to reduce the problem to a SNE well suited to simple solution. In the 80s, the power of the domestic ES computers did not suffice to find the optimal technological parameters on the basis of a mathematical model of chemical manufactory; in such cases, even in abridged versions, the SNEs were solved very slowly.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Currently, an opportunity has appeared to solve in the cloud the problems of global optimization not only in the deterministic case, but also in the conditions of uncertainty, which inevitably arise when the observability of the studied object is low. The amount of computation in such problems increases sharply in comparison with the deterministic setting, and a solution must be found in a multidimensional domain with the dimension greater than the dimension of the original deterministic problem. In the past, the solution of the optimization problems under uncertainty was usually limited to those rare cases when it was sufficient to solve simply the overall problem in terms of its localization to a single process or apparatus. Besides, in the conditions of strong nonlinearity of the constraints, it is practically impossible to obtain the solution of such problems by constructing the analytic convex hulls, and this increases the already high computational complexity. If, even in the conditions of analytically obtained convex hulls, we go to the level of the system of objects (for example, the shop floor), the need for high-performance parallelizing is obvious, as well as the fact that some elements of the computational system are large-block ones.

It should be noted that, when the dimension of the large-block problems of mathematical modeling increases, the understanding of the accuracy of the resulting solution is lost. Can one trust the obtained solution? If, for example, there are used in the calculations the multistage mathematical models of the kinetics of organic synthesis reactions, the reaction rates may vary by some orders of magnitude, and the system of differential equations becomes rigid. To such problems there are reduced the mathematical models of explosive processes, the processes of combustion and firing.

The practice of computing using the software packages of numerical methods gives reason to believe that even in solving the test problems associated with rigid systems, the precision of the floating point representation format of numbers with 16 decimal digits in the mantissa is not sufficient. It is necessary to engage the classes and/or functions for working with the arithmetic of enhanced accuracy. In this case, the time of calculation (particularly, involving standard mathematical functions such as exp(x)) increases dramatically. Even a low-dimensional problem which was before solved easily turns into a challenging computational problem.

It is not always possible to use the now traditional methods of parallelization developed for structurally simple problems (for example, for a number of problems in mathematical physics) with a very large number of similar elements (finite elements, finite differences). On the other hand, in the case of transition to a level, where an individual process vessel becomes an element of any significant system, there arises a tendency to consider a structurally simple element requiring the use of a powerful computing platform (cluster), so again there takes shape a large computing block, whereas the system as a whole becomes large-block.

The character of the parallel computing organization in the new conditions requires, above all, new theoretical justifications and methods of large-block parallelization in the cloud, taking into account the requirements of obtaining a precise and trustworthy solution and the possible presence of global iterative cycles in the calculation system. A motivation for the creation of a new theory is the need to evaluate the computational complexity as a minimization criterion: in the conditions of complicated calculation topologies it is important to correctly distribute the tasks with respect to resources, achieving a significant reduction in the total calculation time.

The proposed methodology of assessing the complexity of large-block high-performance computing is aimed at constructing the architecture of the solution in a cloud, whose resources are used for precise-trustworthy solving the complex problems of mathematical modeling and optimization with the above specific features. The basis of this methodology is the representation of the cloud architecture as a weighted directed graph with the arcs identified with the blocks enlarged by aggregation. The weight of these arcs corresponds to the theoretical complexity of the numerical methods, because the aggregation of the entire set of computational subtasks implies that one block solves one typical problem of computational mathematics. Moreover, the formulas of the complexity estimates should include the size of mantissa, the dimension of the problem, the required accuracy of the solution (if there are present in the local method the iterative cycles such as while (...), for example, the iterative methods for solving SNE). Next, the vertices of the digraph are identified with the cloud servers, distributing the initial data and initiating the calculations in clusters or individual computers (if,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

for example, a university computer network is used) [4].

In this article we use the terminology and notations used in the monograph [5]. The questions of the complexity theory were considered earlier by the authors in [6-9].

The studies are carried out in accordance with the Project No. 1346 from the register of state tasks for the higher education institutions and scientific organizations of the Russian Federation in the field of scientific research.

2. GENERAL THEORETICAL STATEMENTS

We assume that the structure of the problem is known to us from the results of structural identification of the studied system $S^{(0)}$ in accordance with the research goals (in reality, $S^{(0)}$ is the initial system, for example, a shop floor for the acetylene manufacturing; a natural-industrial system of a region etc.); and this structure is a digraph $G^{(0)} = (V^{(0)}, D^{(0)}, \Gamma^{(0)})$, where $V^{(0)}$ is the set of vertices, $D^{(0)}$ is the set of arcs, $\Gamma^{(0)}$ are the weight characteristics of the arcs. Next, $n^{(0)} = |V^{(0)}|$ is the number of vertices, $m^{(0)} = |D^{(0)}| = |\Gamma^{(0)}|$ is the number of arcs of the digraph of a structurally complex computational problem. It is required to find a digraph of the cloud computational system (CCS) for solving the original problem, so that the overall computational complexity becomes minimal:

$$(G^* \equiv (V^*, D^*, \Gamma^*)) = Arg\min_{G}\Theta\left(G\left(G^{(0)}\right)\right),$$

where Θ is an estimate of the computational complexity of the digraph $G = G(G^{(0)})$. The function $G(G^{(0)})$ characterizes the process of structural identification at the stage of the CCS synthesis.

Each of the arcs of the sought-for digraph G identifies a calculation characterized by the following values:

1. Computational complexity $\Theta(d_k)$;
2. The number $T(d_k)$ of digits in the mantissa in the realization of arithmetic operations;
3. The number $I(d_k)$ of input variables that participate in the organization of the iterative process under the condition that this arc is cut.

We should note that in the simplest case (without aggregation of computations) the parametricity $\gamma_k, k = \overline{1,m}$ is a functional of the form

$$\gamma_k = \gamma_k(\Theta(d_k), p(d_k), I(d_k)).$$

We suggest

$$\gamma_k = I(d_k) \times \frac{E_{\Theta,k}+\Theta(d_k)}{\Theta(d_k)} \times \frac{E_{T,k}+[p(d_k)]^{\iota_k}}{p(d_k)},$$

where

1. $E_{\Theta,k} \geq 1$ is a parameter determined by experts, and is a measure of verification of the mathematical description of the simulated object with respect to the original system $S^{(0)}$; for example, a phenomenological model usually has a higher measure of verification than a model constructed according to the "black box" principle and based on approximating the experimental data; into the parameter $E_{\Theta,k}$ one can include, for example, such indicators as the completeness of taking into account the factors and the correspondence to the studied phenomenon of the parameters of the mathematical model; thus, the parameter $E_{\Theta,k}$ expresses the preference for the mathematical description, and if it equals 1, the role of the computational complexity d_k of the arc in its resultant parametricity is minimal.

2. $E_{T,k} \geq 1$ is a parameter determining the increase of parametricity γ_k as the number of digits of the mantissa grows; it is natural to assume that, if for example, we want to compare the parametricity of two arcs, in one of which the number of digits equals 100, and in the other, 50, the latter is preferable for the iteration; on the other hand, a significant influence on the choice of the arc to be cut may have the amount of contours being broken: the bigger it is, the greater is the information load of this arc; we propose to evaluate this parameter as the contourness of the arc d_k;

3. $\iota_k \geq 1$ is an indicator, characterizing the sensitivity of the calculation time to the increase of the number of digits in the calculations connected with the arc d_k; it is defined as the sensitivity degree of the quantity $\Theta(d_k)$ to the increase of requirements for the precision of the obtained local solution in the arc d_k; for example, if in the calculations of arc there is used the Gauss method for solving the systems of linear algebraic equations (SLAE) of dimension n, and, in turn, the greater the dimension n, the more precise, at a principal level, is the calculation of d_k (this kind of problem can occur, for example, in the finite-difference approach to the numerical solution of partial differential equations), then $\iota_k \sim 3$, since, with the increase of the dimension of SLAE, the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

computational complexity of this method is proportional to the cubed dimension of SLAE.

2.1. Basic notations and terms

We will assume that the digraph G is a result of structural identification of a cloud computational system (CCS) S, which is the object of study. The structure of the system is stationary, so the process of its identifying, the structuring $\mathrm{Str}(S)$, is a mapping which associates to the system S a weighted digraph G

$$\mathrm{Str}(S): S \to G. \quad (1)$$

The complexity estimate $\theta(S)$ is an integral quantitative characteristic of the overall computational load of the CCS, an indicator which facilitates ordering of various structures of CCS, proposed from outside, and choosing an option that has minimal complexity, other factors being equal [10].

The functioning of the system S is subject to the objectives of its existence, and this is reflected by an automorphism reflecting the purpose of calculations

$$Aim(S): S \to S. \quad (2)$$

The digraph G is the medium within which there are solved the problems $\mathrm{T(G)} = (\mathrm{T_1(G)}, \dots)$, coordinated with $\mathrm{Aim(S)}$.

Solving the problems from the tuple T(G) is carried out on the scale θ(G) after substituting the complexity estimates of the system S by the complexity estimates of the digraph

$$\theta(G) = des(\theta(S)). \quad (3)$$

which means that "$\theta(G)$ is a characteristic of $\theta(S)$" [5]. Such transition is correct only for the stationary and pseudo-stationary CCS, whose structure remains unchanged in time (or during some time, for the pseudo-stationary ones), or, in other words, for the systems, which are in the state of a classical block-scheme.

The digraph G is a tuple $\mathrm{G} = (\mathrm{V}, \mathrm{D}, \Gamma)$. The tuple G includes

– $V = (v_i, i = 1..n)$ is the tuple of vertices;

– $D = \left(d_k = des\left(v_i \to v_j\right), k = 1..m; i, j \in \overline{\{1, n\}}, i \neq j\right)$ is the tuple of arcs;

– $\Gamma = (\gamma_k, k = 1..m; \gamma_k :=: d_k)$ is the tuple of parametricities; the functor $\gamma(d_k)$ redefines γ_k and is a procedural model of the arc parameterization process, $\gamma_k :=: d_k$ means "γ_k corresponds to d_k".

Further as the text goes, one may encounter the union of an arbitrary tuple with the empty tuple to the right from the concatenation sign: the operation does not change the content of the tuple, i.e. $\forall \mathrm{A} = (\dots), \mathrm{B} = (\dots): \mathrm{B} \Leftarrow \mathrm{A} \cup \emptyset \Rightarrow |\mathrm{B}| = |\mathrm{A}|$. However, if, as a result of concatenation, the empty element is to supplement the content of the tuple, we will use λ-functions in our notations: $\mathrm{B} \Leftarrow \mathrm{A} \cup \lambda\emptyset \Rightarrow |\mathrm{B}| - |\mathrm{A}| = 1$, where $\lambda\emptyset = (\emptyset)$ is a nameless tuple.

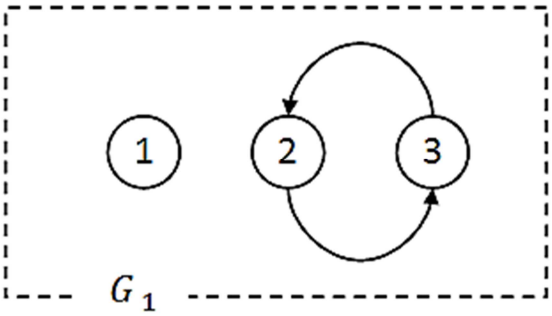
Hereinafter, in the graphic illustrations and some formulas, the vertices "$\mathrm{v_i}$" may be denoted by aliases (auxiliary names) "i".

The CCS digraph G has the following specific features:

1) parametricities are real scalars greater than one: $\gamma_k \geq 1$. Thus, one plays the role of the reference value for parametricities. To the arc $d_k = \left(v_i \to v_j\right)$ there is associated the weight $\gamma_k \in R^{\geq 1}$, $k = 1..m$, where $R^{\geq 1}$ is the set of positive real numbers with the reference point 1.
2) let us confine ourselves to digraphs without isolated vertices and multiple arcs, and this restriction corresponds to the condition

$$\left(\nexists v_i \in V: \nexists j \neq i: \left(v_i \to v_j \vee v_j \to v_i\right)\right) \wedge \nexists k_1, k_2: d_{k_1} = d_{k_2}. \quad (4)$$

The antipodes not satisfying (4) are shown in Figure 1.



Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

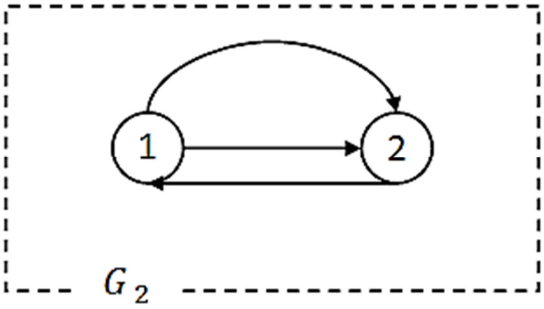


Figure 1. Examples of digraphs for which condition (4) does not hold: digraph G_1 includes an isolated vertex, digraph G_2 includes multiple arcs

2.2. Associative representations of the elements of the CCS digraph

It is important to specify in advance what semantic associations arise in the representation of the CCS structure: what exactly is associated with arcs, and what, with the vertices? In our case, it is most convenient to assign the main computational load to the arcs, whereas the vertices act as servers, performing the distribution of information-computational flows. Figure 2 illustrates the semantic basis of the information-computational load attributed to an arc of the digraph. Note the following:

1) the arc d_k has two possible states: the normal and the cut ones; in the latter the information-computational load includes not only the calculations, but also the functions of controlling the iterative cycle followed by a signal of completion/incompletion of iterations;
2) the information-computational load of the arc in the normal state is determined by the computational complexity of the calculation module $\mathrm{M}_k\left(\mathrm{Z}^{(k)};\mathrm{A}^{(k)}\right)$, where $\mathrm{Z}^{(k)}$ is the vector of the state variables, resulting from the calculations, $\dim\mathrm{X}^{(k)} = q_k$; $\mathrm{A}^{(k)}$ vector of parameters that control the characteristics of computing in the calculation module, $\dim\mathrm{A}^{(k)} = r_k$.
3) the resultant information-computational load of the arc in the condition of cut $\overset{\times}{d}_k := : \overset{\times}{\gamma}_k$ may differ from the ordinary level $\breve{\mathrm{d}}_k := : \breve{\gamma}_k$; the more parameters are included into the vector $\mathrm{A}^{(k)}$, i.e., the larger the number r_k, the greater is the arc's load.
4) selection of the arc for cutting and the subsequent organization of an iterative cycle is performed on the basis of total computational complexity $\gamma_k = \circ\left(\breve{\gamma}_k, \overset{\times}{\gamma}_k\right)$, where $\circ$ is the functional abstractor.

The essence of the information-computational flow is represented by a knowledge frame, it includes the procedural models $\mathrm{M}_\mathrm{k}\left(\mathrm{Z}^{(\mathrm{k})};\mathrm{A}^{(\mathrm{k})}\right)$ and $\gamma_\mathrm{k} = \circ\left(\breve{\gamma}_\mathrm{k}, \overset{\times}{\gamma}_\mathrm{k}\right)$. The information component is represented in the flow by the vectors $\mathrm{Z}^{(\mathrm{k})}$ and $\mathrm{A}^{(\mathrm{k})}$, whereas the computational one, by the above procedural models, and, if possible (if the arc is cut), by a procedural model of the arc cutting in the form of a predicate $\mathfrak{X}_\mathrm{k} = \mathfrak{X}_\mathrm{k}\left(\mathrm{A}^{(\mathrm{k})}, \mathrm{M}_\mathrm{k}\right) \in \{\mathrm{false}, \mathrm{true}\}$. The value false corresponds to continuation of iterations, whereas true, to the end of cycle. In the case when all $\mathfrak{X}_\mathrm{k} = \mathrm{true}, \mathrm{k} = 1..\beta^*$ we take that the computational load in the system equals zero (here β^* is the total number of the arcs being cut).

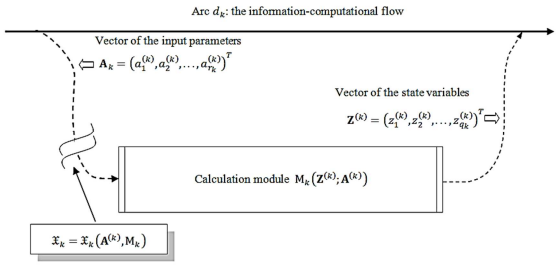


Figure 2. Computational Load Of An Arc Of The Digraph

2.3. Comparability of the CCS digraphs

The structural properties of a CCS digraph are expressed by the collection of two tuples: the tuple of vertices V and the tuple of arcs D. The parametric properties are expressed by the tuple Γ. The structural properties have priority in the graph ranking. It is necessary to introduce appropriate definitions.

Definition 1.

A weighted digraph G is comparable in structure with a non-weighted digraph $\underline{\mathrm{G}}$, this fact being denoted as $\mathrm{G} \rightleftarrows \underline{\mathrm{G}}$, if the weighted adjacency matrix of the weighted digraph G and the adjacency matrix of the non-weighted digraph $\underline{\mathrm{G}}$ are equal up to the sign of the number

$$\left(\begin{Bmatrix} G = con(\mathrm{X}), \\ \underline{G} = con(\underline{\mathrm{X}}) \end{Bmatrix}, \begin{Bmatrix} x_{ij} = des[\gamma_k := :(i \to j)], \\ \underline{x}_{ij} = des[\exists(i \to j) \Rightarrow 1 \nrightarrow 0], \underline{x}_{jj} = sign(x_{ij}) \end{Bmatrix}; i, j = 1..n\right) = true. \quad (5)$$

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where $G = con(\mathrm{X})$ means that "G is rendered concrete by X" [5], ↬ means "else".

Definition 2.

The weighted digraphs G_1 and G_2 are called identical in structure, this being denoted by $\mathrm{G}_1 \sim \mathrm{G}_2$, if

$$\exists \underline{\mathrm{G}}_1 : \mathrm{G}_1 \rightleftarrows \underline{\mathrm{G}}_1, \exists \underline{\mathrm{G}}_2 : \mathrm{G}_2 \rightleftarrows \underline{\mathrm{G}}_2, \underline{\mathrm{G}}_1 = \underline{\mathrm{G}}_2 \Rightarrow \mathrm{G}_1 \sim \mathrm{G}_2 \quad (6)$$

The weighted digraphs may have the same structure and proportionally dependent parametricities.

Definition 3.

The digraphs $\mathrm{G}^{(1)}$ and $\mathrm{G}^{(2)}$, which are identical in structure and are similar in that the parametricities $\mathrm{G}^{(2)}$ can be obtained from the parametricities $\mathrm{G}^{(1)}$ by multiplication by a positive coefficient, are called comparable, this being denoted by $\mathrm{G}^{(1)} \rightleftharpoons \mathrm{G}^{(2)}$, if there holds the condition

$$\left(G^{(1)} = con(\mathrm{X}_1), G^{(2)} = con(\mathrm{X}_2); \left\{x_{ij}^{(1)} = \alpha\, x_{ij}^{(2)}\right\}; i, j = 1..n, i \neq j, \alpha > 0\right) = true. \quad (7)$$

The notation of the form "y=con(x) " means "y is rendered concrete by x".

3. AXIOMATICS OF THE INTEGRAL ESTIMATES OF THE CCS COMPUTATIONAL COMPLEXITY (INITIAL STAGE)

To formalize the estimate $\theta(\mathrm{G})$, we need a number of complexity axioms. They are the theoretical basis of derivation of the formulas for the quantitative evaluation of the CCS complexity.

Axiom 1 (on the estimates of complexity of comparable digraphs).

If $\mathrm{G}_1 \rightleftharpoons \mathrm{G}_2$ and $1 \leq \alpha \leq \overline{\alpha}$, then they are comparable in complexity; moreover, if the complexity $\theta(\mathrm{G}_1)$ is known, then the quantity $\theta(\mathrm{G}_2)$ is proportional to $\theta(\mathrm{G}_1)$, namely

$$\theta(G_2) = \upsilon(\alpha)\theta(G_1), \upsilon(\alpha): \forall \alpha > \circ : \upsilon(\circ) < \upsilon(\alpha) < \upsilon(\overline{\alpha}), \overline{\alpha} > \alpha. \quad (8)$$

In formula (8), the function $\upsilon(\alpha)$ acts as a constant of proportionality, while, in the formulation of the axiom, a restriction in the form of a double inequality is imposed on the value of α.

4. METHOD OF LEXICOGRAPHIC NUMBERING OF THE STRONGLY CONNECTED DIGRAPHS

In accordance with the well-known axioms of complexity due to George Klir [11], the complexity of a system, consisting of a number of subsystems, is not less than the complexity of the entire system. At the same time, it is quite clear that a complexity estimate is constructive only when the George Klir's axiom is fulfilled up to the "equality" sign for isolated subsystems. Therefore, formalization of complexity estimates should be agreed by the method of application to the indivisibility of the system into subsystems, i.e., in the development of estimates the system's digraph should be strongly connected.

Hypothesis 1. If at our disposal we had a way of lexicographic numbering of strongly connected digraphs, then next there would arise a tendency to associate this numbering with the complexity estimates, i.e., to produce thus a digitization of the complexity scale (Figure 3).

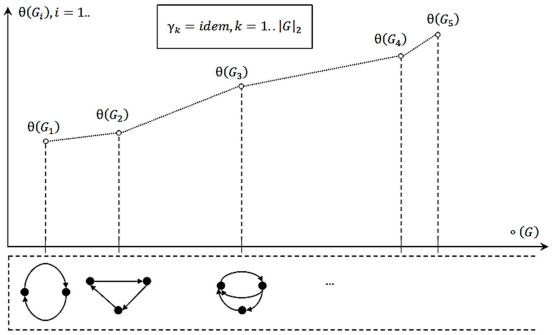


Figure 3. Illustration To The Hypothesis About The Growth Of The Complexity Estimates Of Digraphs Under Their Lexicographic Listing When The Parametricities Are Constant

To substitute the abstractor $\circ(\mathrm{G})$ to a specific function, we associate with the digraph G an invariant $\mathcal{R}(\mathrm{G})$, a unique integer

$$\mathcal{R}(G): \nexists G', G' \neq G \Rightarrow \mathcal{R}(G) = \mathcal{R}(G') \wedge \forall G' \subset G: \mathcal{R}(G) >> \mathcal{R} > (G').$$

The uniqueness of $\mathcal{R}(\mathrm{G})$ consists in the fact that there do not exist two digraphs with the same

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$\mathcal{R}(G)$, and for any subgraph its invariant is always strictly less than the invariant of the digraph to which it belongs. In fact, $\mathcal{R}(G)$ is an estimate of the structural complexity of digraph according to the number of its arcs. Besides the known axioms of complexity, this estimate takes into account a consistent statement: "If a digraph G contains more arcs in comparison with the digraph G′, it is more complex".

The idea of calculating the invariant $\mathcal{R}(G)$ consists in accumulation of the bits starting from the low-order digits, according to the adjacency matrix X, with the subsequent conversion of the bit strings into a non-negative integer.

$$\mathcal{R}(G) = \mathcal{R}_0(G) + \mathcal{R}_1(G),$$
$$\mathcal{R}_0(G) = 0 + \left[\begin{matrix} n & n \\ \dot{} & \dot{} \\ i=1 & j=1 \end{matrix} \left(\begin{Bmatrix} i = j \Rightarrow "" \\ i \neq j \Rightarrow "".x_{ij} \end{Bmatrix}\right)\right],$$
$$\mathcal{R}_1(G) = \sum_{i=1}^{|G|_2} 2^{i+k-1}, k = \left\lfloor \log_2 \mathcal{R}_0(\hat{G}) + \frac{1}{2} \right\rfloor,$$
$$\hat{G} = con(\hat{X}), G = con(X), \dim X = \dim \hat{X}, \hat{x}_{ij} = 1,$$
$$i = 1..n, j = 1..n, i \neq j, n = |G|_1 \quad (9)$$

where the operation "." is concatenation; "" is the empty string; "".x_{ij} is a bit string ("0" or "1", depending on the value $x_{ij} = 0 \vee 1$); the expression"0+bit_string" means that the "bit_string" is transformed into the corresponding positive integer; $\lfloor\ \rfloor$ is the operation of taking the integer part of a number; $\hat{G}$ is the complete digraph with the number of vertices coinciding with the number of vertices of the digraph G.

Figure 4 shows an example illustrating the calculation of $\mathcal{R}(G)$.

(See in the appendix)

Figure 4. An example of a digraph invariant

5. STRUCTURAL DECOMPOSITION OF THE CCS DIGRAPH

In general, the CCS digraph may initially include strongly connected components (bicomponents) or, if they were not present initially, in the process of calculating $\theta(G)$ the digraph G is recursively simplified, and there appear bicomponents. Thus, it is necessary to formalize two radically different states of the CCS digraph:

1. digraph G is strongly connected;
2. digraph G is not strongly connected.

In the first case, the reachability matrix

$$H = (h_{ij})_{n \times n}, h_{ij} = \begin{cases} \exists (i \to \ldots \to j) \vee i = j \Rightarrow 1, \\ \nexists (i \to \ldots \to j) \Rightarrow 0, i, j = 1..n \end{cases} \quad (10)$$

is completely filled by ones; in the second, only partially.

If reachability matrix has at least one zero element, then the digraph G of the system S contains at least two bicomponents corresponding to the strongly connected subsystems. In fact, a parallel is drawn between the concepts of a "strongly connected subsystem" and a "bicomponent" [12]. Figure 5 shows three digraphs: 1) a strongly connected one; 2) a tree of bicomponents; 3) a tree of vertices.

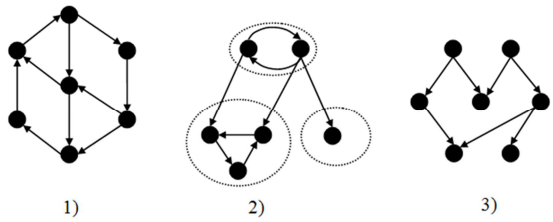


Figure 5. Three variants of a digraph (bicomponents are in the ovals)

Figure 5 illustrates the need to consider three aspects of the formalization of $\theta(G)$:

1) evaluating the complexity of trees of vertices;
2) evaluating the complexity of trees of bicomponents;
3) choosing the simplifying operations to bring a strongly connected digraph to the state of a tree.

6. AXIOMATICS OF INTEGRAL ESTIMATES OF THE CCS COMPUTATIONAL COMPLEXITY: EVALUATION OF THE COMPLEXITY OF TREE-LIKE STRUCTURES

The structure of the tree-like CCS is described in the form of a tree of calculations [13]. A digraph G is called a tree (Remark 1), if it has no contours; and, to separate these digraphs into a separate class, we introduce for them a special notation $\widehat{G}$.

To evaluate the complexity of trees, we formulate the following axioms:

Axiom 1 (complexity of the elementary tree).

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A complexity estimate for the elementary tree with two vertices is no less than the weight of the arc that joins them.

$$\left|\hat{G}\right|_1 = 2, \left|\hat{G}\right|_2 = 1 \Rightarrow \theta\left(\hat{G}\right) \geq \gamma_1, \quad (11)$$

Axiom 2 (complexity estimate for the trees with comparable structures).

If the trees $\hat{G}_1$ and $\hat{G}_2$ are identical in their structure, $\hat{G}_1 \sim \hat{G}_2$, and the total weight of arcs of the tree $\hat{G}_1$ is greater than the total weight of arcs of the tree $\hat{G}_2$, a complexity estimate for the first tree cannot be less than a complexity estimate for the second one

$$\sum_{k=1}^{|\hat{G}_1|_2} \gamma_{1,k} > \sum_{k=1}^{|\hat{G}_2|_2} \gamma_{2,k} \Rightarrow \theta\left(\hat{G}_1\right) \geq \theta\left(\hat{G}_2\right). \quad (12)$$

Axiom 3 (on the relation between the complexity of a tree and a subtree).

The complexity of any subtree $\hat{G}' \subset \hat{G}$ is less than the complexity of the tree which contains it

$$\hat{G}' \subseteq \hat{G} \Rightarrow \theta\left(\hat{G}'\right) \leq \theta\left(\hat{G}\right). \quad (13)$$

In accordance with the adopted axioms, two variants of formulas are proposed for estimating the complexity of trees, whereas each option is made agree with the set of axioms (11)-(13): Variant # 1 – evaluation of the tree complexity by the total weight of the arcs

$$\theta^{(1)}\left(\hat{G}\right) = \sum_{k=1}^{|\hat{G}|_2} \gamma_k, \quad (14)$$

Variant # 2 – evaluation of the tree complexity while taking into account the load of the paths

$$\theta^{(2)}\left(\hat{G}\right) = \sum_{k=1}^{p} \sum_{j=1}^{|\mathbb{P}_k|} \hat{\gamma}\left(\mathbb{P}_{k,j}\right), \quad (15)$$

where p is the total number of all possible paths from the vertices, belonging to the set of exogenous vertices $\overset{V}{\rightarrow}$ of the tree $\hat{G}$, to the vertices, belonging to the set of endogenous vertices $\overset{V}{\leftarrow}$ (see the illustration in Figure 6).

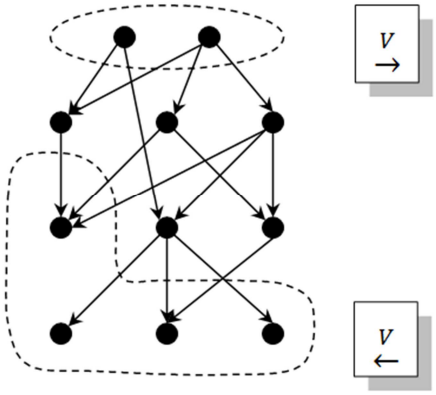


Figure 6. Illustration for the formula (15)

The selection of a criterion for the complexity evaluation of the CCS with a tree-like structure depends on the nature of calculations [14]:

1) A CCS S is a tree-like calculation module executed by *independent computing devices*. In such cases the estimate $\theta^{(1)}\left(\hat{G}\right)$ is suitable: it is an upper bound of computational complexity. On the other hand, this estimate can be used also in the case of distributed calculations with *dependent computing devices*, but not as a complexity estimate, but as an estimate of the overall computing capacity of CCS; this estimate can be used in designing the parallelization schemes: the smaller the computing capacity, the lower is the cost of the calculations themselves; there arises the problem of minimizing $\theta^{(1)}\left(\hat{G}\right)$ on the set of variants of the parallelization schemes.

2) The estimate $\theta^{(2)}\left(\hat{G}\right)$ can be applied to the cases where CCS is a collection of computing devices with a tree-like structure, whereas the main computational load is not on the arithmetic operations, but on the transmission of large volumes of information; a complexity estimate is proportional to the total loading of channels by the information flows during the stable period with a constant structure of calculations; the larger the average amount of information per a channel, the more loaded it is and the greater is the contribution of this channel to the overall complexity evaluation for the entire tree-like structure; for such cases, formula (15) is appropriate.

7. COMPLEXITY ESTIMATE FOR THE HIERARCHIC CCS

The hierarchic CCSs differ from the tree-like CCSs by that bicomponents play the part of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

vertices. From the viewpoint of the graph theory, the structure of the hierarchic CCS is a tree of strongly connected subgraphs (SCS). To evaluate the complexity of the trees of SCSs (abbreviated as TSCS), it is necessary to convert TSCS into a tree of generalized vertices, a Hertz digraph, find the weight of the generalized arcs and apply to the obtained result the formulas for estimating the complexity of the trees of vertices [15-16].

Previously we adopted the concept that the complexity of calculations, concentrated in an arc, is associated with the weight of the arc, whereas the vertices are identified with the nodes of data distribution (data servers) (Remark 2). Extending this assumption to TSCS, it is quite logical to associate the complexity of an individual SCS with the weight of a generalized arc coming from the SCS. In the case when the SCS has several generalized arcs coming from it, the weight of each of them is reduced by as many times as there are generalized arcs coming from the generalized vertex.

On the conceptual level, to go from a tree of SCSs to a tree of vertices, we need a method of conversion of a tree of SCSs into a Hertz digraph, and this will require:

1) a method of unique concretization of SCS;
2) a method of evaluation of the SCS complexity;
3) a method of weighing the generalized arcs.

As a result, there will be formed a tree of generalized vertices, the complexity of which is evaluated according to the rules formulated earlier in the section 6.

7.1. Method of unique concretization of SCS

We need to construct a quasi-triangular form of the adjacency matrix $\ddot{X}: \ddot{X} = con(G), \ddot{X} := : \underline{G}, G \rightleftarrows \underline{G}, \ddot{X} := : X$, where X is the traditional non-weighted adjacency matrix of the initial digraph G. To this end, we turn to the method of constructing procedural models on the basis of operator equations.

A procedural model of construction a quasi-triangular form $\ddot{X}$ of the adjacency matrix X is verified by checking the following condition (which is illustrated in Figure 7):

$$\begin{gathered}①: \lambda(\beta) \equiv \left\{\left(bic\left(\tilde{G}_i \setminus \mathfrak{D}_i\right) = true, i = 1..\beta\right): \left(\bigcup_{i=1}^{\beta} \left(\tilde{G}_i \cup \mathfrak{D}_i\right) = G\right)\right\} = true \ \wedge \\ \forall \beta_1 < \beta: \lambda(\beta_1) = 0, \mathfrak{D}_i := : \tilde{G}_i, \mathfrak{D}_i = \left(d_j^{(i)}, j = 1..Y_i\right), \ddot{X}^T := : \bigcup_{i=1}^{\beta} \left(\tilde{G}_i \cup \mathfrak{D}_i\right), \\ ②: \tilde{G}_{i_1} \prec \tilde{G}_{i_2} \prec \ldots \prec \tilde{G}_{i_\beta}: \forall j \in \overline{\{1, \beta - 1\}}: \nexists l > j: \exists k, \left[\left(d_k = \left(v_{k_1} \rightarrow \circ\right)\right): v_{k_1} \in \tilde{G}_{i_l}\right],\end{gathered} \quad (16)$$

where $\lambda(\beta)$ is a λ -predicate, $bic(G)$ is the predicate equal to *true*, if digraph G is strongly connected. The usage of a λ-predicate in this case if justified by that the first condition contains one and the same predicate twice, and it is the only one in this condition.

(See in the appendix)
Figure 7. Illustration to the condition (16)

If a procedural model of constructing the matrix $\ddot{X}$ and the SCS $\tilde{G}_i$, corresponding to this matrix together with the tuples of outgoing arcs $\mathfrak{D}_i, i = 1..\beta$, is written in the form of a conventional algorithm, then verification of the condition (16) is carried out as follows:

- there is formed a sufficiently representative (Remark 3) selection of initial digraphs $G: bic(G)$;
- to all paragraphs of this selection the procedure is applied of identifying the SCS, and the result is stored in an array of tuples;
- each of the elements of the resulting array is checked for the conditions ① and ②;
- if it appears that both two conditions are met for each element of the array, the conclusion is made about successful verification.

In Figure 8 there is given an example of construction of a tree of SCSs. In this example the digraph is split into three SCSs $G = con(\tilde{G}_1 \prec \tilde{G}_2 \prec \tilde{G}_3)$, whereas the corresponding tree has two aggregation functors: $\chi_1(\gamma_{1,k_{1..3}}) \equiv \chi_1(\gamma(2 \rightarrow 3), \gamma(2 \rightarrow 5), \gamma(7 \rightarrow 6))$ and $\chi_2(\gamma_{2,k_{1..4}}) \equiv \chi_2(\gamma(3 \rightarrow 1), \gamma(5 \rightarrow 4), \gamma(6 \rightarrow 1), \gamma(6 \rightarrow 4))$.

(See in the appendix)
Figure 8. Illustration to anti-lexicographic sorting of the vector $\mathcal{H}$

It should be noted that the SCS resulting from the anti-lexicographic sorting of the bit strings, the components of vector $\mathcal{H}$, form a hierarchical structure.

A procedural model of searching SCS is a map $Dec: G \rightarrow (\tilde{G}_i, \mathfrak{D}_i, i = 1..\beta)$. We propose the following procedural model for searching SCS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$Dec \triangleq \left(\begin{array}{l} \mathbb{I}: 0 + \mathcal{H}_{\mathbb{I}_j} \geq 0 + \mathcal{H}_{\mathbb{I}_{j+1}}, j = 1..n-1 \Rightarrow \lambda(k) = 0 + \overset{n}{\underset{j=1}{\dot{}}} h_{kj}, \\ \bar{G}_i = \left(\begin{array}{l} \left(\begin{array}{l} \forall k \in \mathbb{I}_{L\setminus G_i}: \lambda(k) = idem; \\ l = 1 + \left[(0 := (i = 1)) \vee \sum_{j=1}^{i-1} |\bar{G}_j| \right] \end{array} \right)^{\lambda} \\ \mathfrak{D}_i = \overset{\circ}{\bigcup_{k=1}} \left[G\left(\left(v(\underline{d}_k), v(\overline{d}_k) \right), (d_k), (\gamma(d_k)) \right) : \underline{d}_k \in \bar{G}_i, \overline{d}_k \notin \bar{G}_i \right] \end{array} \right), i = 1..\beta \end{array} \right) \quad (17)$$

Consider the expression (17). The result of calculations in the procedural model Dec is a permutation $\mathbb{I}[1..n] = (\mathbb{I}_1, \mathbb{I}_2, \ldots, \mathbb{I}_n)$ of the segment of natural numbers from 1 to n. If, following this permutation, we also simultaneously rearrange the rows and columns of the transposed adjacency matrix, we obtain a quasi-triangular form $\ddot{X}$, on the basis of which we find the SCS $\widetilde{G}_i$, along with a set of tuples of outgoing arcs $\mathfrak{D}_i, i = 1..\beta$. The condition of comparison of the elements, sorted in the anti-lexicographic order, is reflected by the inequality $0 + \mathcal{H}_{\mathbb{I}_j} \geq 0 + \mathcal{H}_{\mathbb{I}_{j+1}}, j = 1..n-1$. The ending of the strings. $\left(\begin{array}{c} \varsigma(n) \\ "" \overset{\frown}{+} (n-i+1) \end{array} \right)$ in the vector $\mathcal{H}$ serves for lexicographic numbering of vertices within the SCS, in accordance with the numbering of the vertices in the original digraph. Recall that the expression 0+ "bit_ string" is equal to an integer obtained from the direct binary code recorded in the bit string from left to right, starting from the most significant bit.

Figure 9 shows the result of sorting, it corresponds to the digraph in Figure 8. It is shown how the ending of the bit strings in the vector $\mathcal{H}$ helps building the numbers of vertices in the framework of SCS according to ascending of indices: for example, $\widetilde{G}_1$ is characterized by the vertices (v_2, v_7) rather than (v_7, v_2).

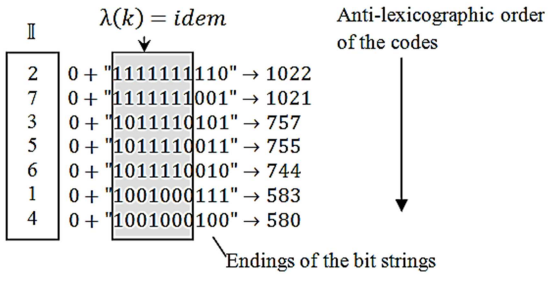


Figure 9. Illustration to the procedural model (17)

Note that, according to the above principles, to find SCS it is not necessary in the detection process to find the contours of the digraph, one only needs to know the reachability matrix.

7.2. Method of searching SCS in the presence of cutpoints in the CCS graph

In some cases, evaluation of the computational load using the complexity estimates is made difficult by the petal topology of CCS, when the central place in the system is assigned to the cloud server with the protected data, or when a separate subsystem is formed by the cloud server, directly connected, at the same time, with several computing substations with a strong connected structure and, possibly, clusters. Figure 10 illustrates in a general form a grid-system aimed at solving a hypothetical problem with global iterations [17]. Consider the CCS digraph in Figure 11. It is shown without the cut arc corresponding to the global iteration cycle. The problem of structuring in which the cloud server, along with computing substations, is combined into a single bicomponent, is that all three elements are considered as a whole; the iteration cycles may contain undesirable association of the cloud server with the vertices belonging to computing substations, as shown in Figure 12. The variant b) is different in that, as a result of decomposition, there are found two contour SCSs instead of one component as in the variant a), these are $\dot{G}_1$ and $\widetilde{G}_2$. The working of the computing substations corresponding to these SCSs, is now controlled by two iterative cycles Ts1 and Ts2, and, at that, the cutpoint CP does not lose its significance as the coordinator of computations. In this case (Remark 4) the complexity estimate is equal to the total complexity $\theta(\dot{G}_1 \cup \dot{G}_2) = \theta(\dot{G}_1) + \theta(\dot{G}_2)$.

In some cases, the cloud server can be regarded as an independent subsystem as shown in Figure 13, then the number of cutpoints increases. If the cloud server is connected to two substations, it is natural to assume that there are exactly two cutpoints in the $\dot{G}_3$ subsystem, i.e., their number is equal to the number of substations associated with the server.

The situations, when in structuring the CCS there appear the cutpoints in the resulting digraph, are possible in the organizations of grid systems within the large network information systems with several clusters, which are geographically distant from each other, when the backbone lines are loaded quite heavily, being the lines of computer

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

communication of the general geographic scope [18-19].

Note also that, as a cloud server, there can be, for example, the branch server. With regard to the problems of organization of high-performance cloud computing, such situations are possible, for example, when solving systems of equations of large dimension, divided into two (or more) blocks with one (or more) equation on the separation boundary (the equation of conjugation [20]). In such cases, one can find a permutation of the rows and columns of the adjacency matrix, applying which this matrix acquires a pseudo-quasi-diagonal form, as shown in Figure 14.

Generally speaking, bicomponents and the contour subgraphs may be considered strongly connected subgraphs of different types, and, on formal grounds, in a digraph either types can exist simultaneously. However, a differentiated approach to the enumeration of subgraphs (with their subdivision into types) significantly complicates the further formalization. It is reasonable to equate bicomponents, in which there are no cutpoints, with the contour subgraphs if we use a bouquet model of decomposition. The choice of the specific procedural model of the digraph decomposition into SCSs depends on the specific situation in which the complexity is evaluated by gradual simplification of the system under study with the further composition of the estimates of primitive structures, going up from the lower level of decomposition. With respect to cloud computing, one of the factors in favor of the method of contour subgraphs is the original structure of connecting of cloud servers. For example, if there are connections of the "star" type in the scheme and the problem is set of evaluating the stability of working of the cloud system, then the contour subgraphs are preferred over bicomponents.

(See in the appendix)
Figure 10. Example of a grid system with global iterations and a cloud server

(See in the appendix)
Figure 11. Structuring of a grid system: the case of one bicomponent

(See in the appendix)
Figure 12. Variants for decomposition of the CCS digraph in Figure 11:
a) without taking into account a cutpoint; b) taking into account a cutpoint

(See in the appendix)
Figure 13. The variant of structuring, when the cloud server is represented as a contour SCS

(See in the appendix)
Figure 14. General form of the adjacency matrix for a system of equations of large dimension with the equation of conjugation of blocks

8. THE METHOD OF STRUCTURAL-PARAMETRIC MINIMIZATION OF THE CCS DIGRAPH

A CCS digraph may contain the elements that contribute to an increase in computation time of the complexity estimation, which adversely affects the application of these estimates in the online regime. Among these elements, hindering the analysis of complexity, there are the branches of calculations performed serially and hidden parallel branches of calculations. In Figure 15 there are shown: an example of the digraph (G), including redundant elements, and the result of structural-parametric minimization (SPM), the digraph G'.

(See in the appendix)
Figure 15. Example of a CCS digraph with redundant elements

Transitive computational flow $2 \rightarrow 4 \rightarrow 5$ (the arcs are highlighted by thick lines) is replaced by the generalized arc $2 \Rightarrow 5$, its parametricity is shown in Figure 15 in the form of a question mark: it is required to determine how it will be evaluated. Through the generalized arc (such arcs are marked by double lines) $2 \Rightarrow 5$ there goes computational flow, which is performed parallel to the computational flow in the initial arc $2 \rightarrow 5$ of the CCS digraph (it is marked by circles and by dotted line connecting them): these arcs have the same initial and end vertices. The parametricity of the generalized arc of the digraph G' is also shown in the form of a question mark. Evaluation of the complexity of parallel computational flows should take into account the character of their carrying out: if these flows are processed in parallel, then the complexity evaluation must be coordinated with the critical line, whose complexity is maximum.

There is another important point, which is illustrated in Figure 16: after replacing the transitive branch by a generalized arc, a reversing

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

loop should not appear in the digraph G'. In the example in Figure 16, the transitive branch 1→2→3→6 is replaced by the generalized arc 1⇒6; there is located parallel to it the transitive branch 1→4→5→6. As a result, there is formed a generalized arc, which, together with the initially existing arc 6→1, can be reduced on the basis of transitivity to one arc, making it a loop. However, we have previously assumed that the digraphs do not contain loops, therefore, as a result, there is obtained a digraph G', in which the vertex 6 is a cutpoint.

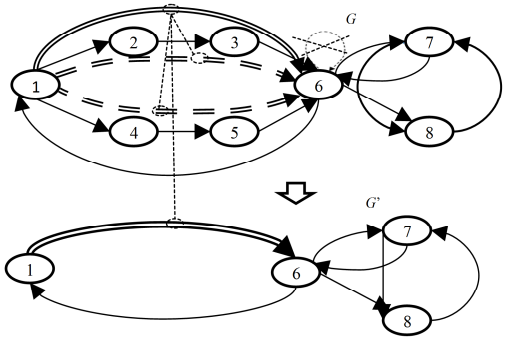


Figure 16. Illustration of inadmissibility of loops in the contraction of transitive computation branches

Structural-parametric minimization (SPM) on the level of concepts can be divided into two levels: a) the upper, the structural one and b) the lower, the parametric one. The upper level is already designated, the lower is designated partially, for the transitive branches and, at the conceptual level, for the hidden parallel branches. Let us formulate a procedural model of the SPM method on the recursive basis.

1. The input StrMin(G): $G = (V, D, \Gamma)$ is a CCS digraph; $\breve{\mathrm{C}}(G) = \left(\check{c}_{ij}(G)\right)_{K\times m}$ is the classical matrix of contours, corresponding to the digraph G.
2. Initialize the return value StrMin, the digraph $G' \Leftarrow G$.
3. Make zero the tuple of arcs, which constitute the transitive branch: $D^* \Leftarrow \emptyset, m^* \Leftarrow \emptyset$.
4. Initialize the Boolean array $pD^* = \left[pD_j^* \Leftarrow false, j = 1..m\right]$. If $pD_j^* = true$, then the column with the number j is treated as "already considered" and omitted.
5. A cycle over the "non-considered" columns of the matrix $\breve{\mathrm{C}}(G)$, $j = (1..m) \wedge pD_j^* = false$.
6. A cycle over the remaining columns of the matrix $\breve{\mathrm{C}}(G)$, except for the column number j: $\nu = 1..j-1..j+1..m \wedge pD_\nu^* = false$.
7. Check the sameness of the columns $\check{c}_{ij} = \check{c}_{i\nu}, i = 1..K$? If they are identical, then add an arc into the tuple $D^* \Leftarrow D^* \cup \left\{d^*_{++m^*} \Leftarrow d_{D^*=\emptyset\Rightarrow j\nu}\right\}$ and indicate the corresponding column as "already considered": $pD^*_{D^*=\emptyset\Rightarrow j\nu} = true$.
8. End of the cycle over ν.
9. End of the cycle over j.
10. Test the obtained tuple in terms of transitivity $\xi(\mathrm{D}^*) = true$? If the testing came to be successful, proceed to item 12.
11. Apply the functor $\breve{\xi}[\mathrm{D}^*]$: if $\breve{\xi}[\mathrm{D}^*] \neq \emptyset$, then we put $\mathrm{D}^* \Leftarrow \mathrm{D}^* \setminus \breve{\xi}[\mathrm{D}^*]$, otherwise, the procedural model finishes its work.
12. Delete the transitive branch from the resulting digraph $\mathrm{G}' \Leftarrow \mathrm{G}' \setminus \lambda G(\circ, \mathrm{D}^*, \circ)$ and find a generalized arc d^* from the solution of the operator equation (41), then add information into the resulting digraph $\mathrm{G}' \Leftarrow \mathrm{G}' \cup \lambda G(\circ, \{d^*\}, \circ)$.
13. Check the presence of a hidden parallel branch, initiate a cycle of exhaustive searching the arcs of the digraph G', all except for d^*, $k = 1..|G'|_2, d_k \neq d^*$.
14. The arc d_k includes parallel computations with the generalized arc d^*? $\mathcal{G}(d^*, d_k) = true$? If "yes", then glue the arcs $d^* \Leftarrow d^* \pm d_k$.
15. End of cycle over k.
16. Check $\mathrm{G}' \neq \mathrm{G}$? If "yes", then it is necessary to continue recursion: $\mathrm{G}' \Leftarrow \mathrm{StrMin}(G')$.
17. The output is the digraph G'.

9. COMPLEXITY ESTIMATES OF THE STRONGLY CONNECTED COMPUTATIO-NAL PRIMITIVES

Earlier in sect. 2, an assumption was made about the transformation of the estimates of computation load of CCS into the complexity estimates for the objects of the "digraph" class. As a consequence, in this section the term "strongly connected computational primitive" will be replaced by the term "the simplest strongly connected digraph". "The simplest" is understood in the sense that the formulas of the complexity estimates are derived without using any special algorithms or programs.

Evaluation of the digraph complexity presupposes gradual simplification, down to such primitive structures, for which the estimate of θ(G) is derived as a formula in advance. Thus, in the process of simplification, instead of repeated

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

analysis of the previously analyzed structures, there are used the accumulated knowledge with the ready answers. The first in the list of the tested and ready to be included into the database knowledge there should be strongly connected digraphs of small dimension, consisted of 2 and 3 vertices. For them, we need to derive the complexity estimation formulas.

The criterion $\mathcal{R}(\mathrm{G})$ (formula (9)) allows ordering the list of strongly connected digraphs, which are representation of the simplest topology of iterative and cyclic calculations. We need to remove from this list isomorphic digraphs, and, for the remaining digraphs, to find the estimation formulas for the complexity θ (G) in the form of absolute predicates.

It should be noted that in the criterion $\mathcal{R}(\mathrm{G})$ the weight of the arcs is not used and $\forall \mathrm{G}: \mathrm{G} = \underline{\mathrm{G}} \Rightarrow \mathrm{G} \Leftarrow \mathrm{G}\backslash\Gamma$. Beyond the issues of indexing of the list elements, in this section we will use the restored $\mathrm{G} \Leftarrow \mathrm{G} \cup \Gamma$.

Consider the codes of all possible strongly connected digraphs with three vertices, ranking them in the order of increasing of the criterion $\mathcal{R}_0(\mathrm{G})$: 23, 25, 27, 29, 31, 38, 39, 45, 46, 47, 54, 55, 57, 58, 59, 61, 62, 63. Due to its uniqueness, the criterion $\mathcal{R}_0(\mathrm{G})$ is taken as the basis of constructing the isomorphism diagrams for the class of strongly connected digraphs with the given vertex dimension $|\mathrm{G}|_1$. The arrows in these diagrams go from right to left, coming to the codes of the basic digraphs. Figure 17 shows the results of studying the isomorphism of the above-mentioned list of digraphs according to the criterion $\mathcal{R}_0(\mathrm{G})$, in the form of a diagram of isomorphisms. A failure with respect to the arc dimension $|\mathrm{G}|_2$ is observed immediately, starting with the code 23: there corresponds to it a digraph with 4 arcs, whereas to the code 25 there corresponds a digraph with three arcs. There is clearly a violation of the principle of assigning the invariants (9). In Figure 17, the basic codes are marked in gray, i.e. the corresponding isomorphic digraphs have the code $\mathcal{R}_0(\mathrm{G})$, the value of which exceeds the basic code.

(See in the appendix)

Figure 17. The codes $\mathcal{R}_0(G)$ of digraphs with three vertices and the number of arcs (below)

10. COMPLEXITY OF A DIPOLE

A dipole is a strongly connected digraph; let us adopt for it the notation of the form $\overset{(3)}{\mathrm{G}}$, where the superscript "3" is the value of the invariant $\mathcal{R}_0\left(\overset{(3)}{\mathrm{G}}\right)$. There are two vertices and two arcs in the dipole:

$$\overset{(3)}{G} = (V, D, \Gamma), V = (v_1, v_2), D = (d_1, d_2), \quad d_1 = (v_1 \to v_2), d_2 = (v_2 \to v_1), \Gamma = (\gamma_1 :=: d_1, \gamma_2 :=: d_2). \tag{18}$$

The dipole $\overset{(3)}{\mathrm{G}}$ is shown in Figure 18. It is a mathematical description of the structure of the simplest CCS. The dipole vertices correspond to the blocks of the information distribution (servers), whereas the arc, to the computing resources (computers, clusters, cloud platforms) [21].

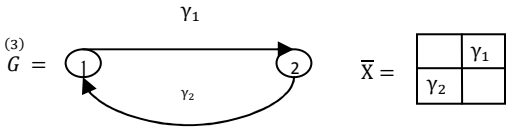


Figure 18. A dipole and its weighted adjacency matrix

Evaluation of the complexity of systems is produced by a series of simplifications implemented in the same way. In the cognitive subtext, a series of simplifications can be represented as a tree of recursion when calculating the complexity estimates, as it is demonstrated in Figure 19.

The structure of the initial state of CCS is represented by a strongly connected digraph $\mathrm{G} \neq \overset{(3)}{\mathrm{G}}, \mathrm{bic}(\mathrm{G}) = \mathrm{true}$. It is required to estimate the computational complexity of CCS. For this purpose, the initial digraph undergoes gradual decomposition. The initial state of the digraph is assumed to be the zero stage. With regard to CCS, it is the dipole that is the final structure in the tree of recursive calculation of the complexity estimate.

In the process of evaluating the complexity of large-block calculations with global iterations it is a dipole that is a crucially important CCS. Finding a way to transform a dipole into the disconnected state, and simultaneously making the complexity evaluation, one can evaluate the complexity of the entire CCS. At that, in addition to the dipole one must be able to evaluate the complexity of the trees of calculations (see sect. 6).

(See in the appendix)

Figure 19. An example of tree recursion: G is the CCS digraph in its original state; $\mathrm{B} = 3$ *is the number of*

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

levels of hierarchy; $\beta_i, i = 0..B$ *is the width of the levels in the tree of recursion*

Returning to Figure 18, it is easy to notice the presence of two alternatives for disconnecting a dipole: either a break of the arc (1→2), or a break of the arc (2→1). In CCS a break of any arc is connected with the need to iterate with respect to one or more variables, transmitted as a result of the information distribution at the vertices for the subsequent calculation in the arcs of digraph. To resolve the alternative, it is necessary to take into account that the arcs are a topological representation of the computational process, and the choice of the arc, along which the iteration is to be performed, is determined by the complexity of arcs and their articulation in the CCS digraph. We should at once make a reservation that, before calculating the overall evaluation of the complexity of the entire CCS, the parametricities of all arcs of the digraph, without exception, should to be known beforehand.

Procedurally, a computing dipole can be conveniently presented as a CCS with alternative choice of the iteration order: either the cut of the arc d_1 is a procedural realization of sect. 2, or, conversely, the arc d_2 acts as such. Figure 20 shows a graphical interpretation of the procedural model of a dipole as an iterative CCS. A dipole has "input" and "output": at the input there is performed taking off the information, the vertex v_1 is marked by the sign "-", whereas the vertex v_2, at the output by the sign "+", respectively: it is the returning of information. In this case, the dipole can be considered as non-automorphic digraph.

$$G^{(3)} \nsim G'^{(3)}, G^{(3)} = (\{v_1, v_2\}, \{d_1 = (1 \to 2), d_2 = (2 \to 1)\}), \quad G'^{(3)} = (\{v_2, v_1\}, \{d_1 = (2 \to 1), d_2 = (1 \to 2)\}). \tag{19}$$

In the procedural block "a" there acts the mapping w_1, producing transformation of the original information, the vector of tuples x_0 into a subvector $x_1^{(2)} \sqsubset x_0$; there takes place the refinement of the composition of the initial approximation with respect to the reverse calculation flow coming from the "+" -vertex of the dipole to its "-" -vertex. The switch Sw0 changes the course of setting the initial approximation and, thus, the order of iteration cycles. In the "on" position, the initial information is sent to the procedural block "a", whereas in the "off" position, the initial information, the vector of tuples x_0, is sent to the vertex of the dipole with the indication "-", into the procedural block "b", where the selection of information for the vertex 2 is carried out by the mapping w_2.

(See in the appendix)

Figure 20. Dipole as a procedural model of iterative CCS

The main computational load falls on the blocks "c" and "d". There act the maps ϕ_1 and ϕ_2, the calculations are carried out in the direction from the vertex labeled by "-" to the vertex labeled "+" (ϕ_1) and, vice versa (ϕ_2). In the comparators "e" and "f", there is performed the calculation of the predicates $U^{(1,2)}(\bullet)$ and, if the predicate is true, then the calculations are finished, "Exit". Otherwise, the information, previously obtained from the procedural blocks "b" and "c", is passed to the vertices of the dipole for further iterations. On the way from the vertex to the procedural block there are the switches Sw1 and Sw2 (the down arrow means "the way is closed", the up one, on the contrary, means "the way is open"). They control the operation of the dipole; in any case, there operate either one of the comparators or both. It is necessary to determine how exactly the comparators are set, there depends on it the overall computational complexity of the dipole: it should be minimal; an estimate of the total complexity of the CCS digraph is built on this.

An estimate of the dipole complexity includes two quantities:

1. Complexity of the simplification procedure θ_s, as a result of which the dipole becomes a tree (the index "s" stands for "simplification");
2. Complexity of the arc θ_r, obtained as result of simplification (the index «r» stands for "residual", i.e. θ_r is the residual complexity).

Let us formulate the axiomatics. Taking into account that in the working of dipole there are possible two variants of turning on the comparators, then the axioms are also two.

Axiom 5 (complexity of the dipole with one comparator).

The complexity of the dipole in the conditions when only one of the computational blocks bypasses the comparator does not exceed the total complexity of the entire dipole as a single system, namely,

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

$$\theta\left(\overset{(3)}{G}\right) \geq \theta_{sr} \equiv \theta_s + \theta_r. \qquad (20)$$

Axiom 6 (complexity of the dipole with two comparators).
In the general case, the estimate of the dipole complexity is calculated as a minimum of two estimates: the complexity estimate for iterations, resulting from the cut of the arc 1→2 and a similar estimate for the arc 2→1

$$\theta\left(\overset{(3)}{G}\right) = \min\{\gamma_1(1+\gamma_2), \gamma_2(1+\gamma_1)\}. \qquad (21)$$

It should be noted that in the future the formula (21) will have to be concretized for the catalog of typical blocks associated with the computational methods. For the non-typical blocks it is necessary to obtain the experimental characteristics: the parametricities γ_1 and γ_2 as the functions of the calculation accuracy and the characteristics of the dimension of the problems being solved. In the Figure 21 there is presented the experimental dependence of the calculation time for the Gauss method of solving the systems of linear algebraic equations as a function of the dimension n of the system and the number p of digits of the mantissa p.

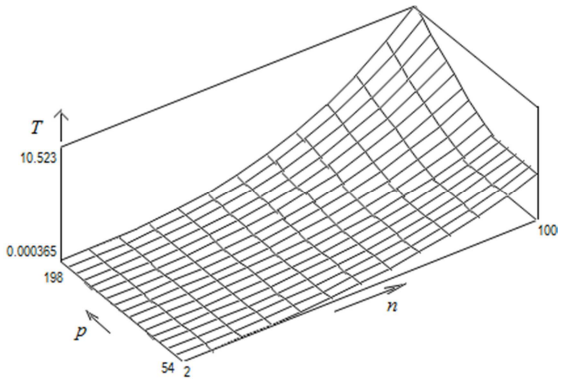


Figure 21. Dependence of the calculation time for the system of linear algebraic equations (sec.) on the dimension of the system and the size of the mantissa

To approximate the experimental dependence shown in Figure 21, we used the dependence $T(n,p) = \alpha n^3 p^2 + \beta n^3 p + \gamma n^2 p^2 + \delta n^2 p$. Particularly, for the Gauss method we obtained the following values: 10-10×($\alpha = 1.88, \beta = 59.1, \gamma = 10.4, \delta = 184$). Similar studies were carried out for a number of methods for solving linear algebraic systems, the approximation coefficients were found. The obtained data led to the conclusion about the applicability of formula (21): the maximum deviation in the case, when in the dipole arcs there participated the solution of linear algebraic systems, did not exceed 5% of the relative error.

11. PROCEDURAL MODEL FOR EVALUATING THE COMPUTATIONAL COMPLEXITY OF CCS

One of the outcome of the development of a methodology for assessing the computational complexity of CCS is a procedural model, which takes into account all aspects of the developed theory.

1. Input (recursive adapter): digraph $G = (V, D, \Gamma)$.
2. Is the digraph strongly connected? $bic(G) = 1$? If "yes", then go to item 12.
3. Structural decomposition: subdivide G into $SCS_i, i = 1..\beta$.
4. $\theta_0 = 0$.
5. Cycle over SCS_i, $i = 1..\beta$.
6. Input (recursive adapter): digraph G = SCS_i.
7. $\theta_0 = \theta_0 + \theta$.
8. End of the cycle over *i*.
9. $\theta = \theta_0$.
10. Output: θ.
11. Up to isomorphism, is the digraph G present in the knowledge base of complexity estimates (the knowledge base is indexed by invariants of strongly connected digraphs)? If "Yes", then make an inquiry into the knowledge base, obtain an estimate $\theta \equiv \theta(G)$ and exit the recursive adapter with the value θ.
12. Carry out SPM of the digraph $G = \overrightarrow{\min}(G)$.
13. Construction of the matrix of contours $\breve{C}(G)$.
14. $\theta_0 = HUGE\_VAL$ (the largest of all possible real numbers in the C++ language).
15. Cycle over the arcs of the matrix of contours, $i = 1..m$.
16. Cut of the arc $G' = G\backslash\{d_i\}$.
17. Input (recursive adapter): G = digraph G'
18. Refine the complexity estimate $\theta = \gamma_i(1+\theta)$.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

19. $\theta < \theta_0$? Is the estimate decreasing? If "NO", then exit the cycle over *i*.
20. Store the estimate $\theta_0 = \theta$.
21. End of cycle over *i*.
22. Exit with the value $\theta = \theta_0$.

12. CONCLUSION

A theoretical basis is developed for evaluating the complexity of large-block cloud computing, using the arithmetic with enhanced accuracy, which includes the methods and procedural models intended for designing CCSs. Currently, the work is underway towards the further elaboration of the complexity estimates for real-world computing dipoles. The computational experiments are carried out on the use of the arithmetic with enhanced accuracy in various numerical methods with the further approximation of the obtained results of testing. It is planned to obtain specific expressions for the parametricity functionals of the arcs of the CCS digraph in solving a number of large-block problems of mathematical modeling.

REFERENCES

[1]. Ostrovsky, G.M., 1975. Optimization of Complex Chemical-Technological Schemes. Moscow: Khimiya.

[2]. Ostrovsky, G.M., 1980. Decomposition of Complex Chemical-Technological Schemes. Moscow: Khimiya, 1980.
[3]. Hansel, K., Yu.M. Volin and G.M. Ostrovsky, 1980. Methods of Structural Analysis in the Problems of Studying Chemical-Technological Schemes. Vol. 4 (89). Moscow: NIITEKHIM.

[4]. Zykov, A.A., 2004. Fundamentals of the Graph Theory. 3rd Ed. Moscow: Vuzovskaya Kniga, pp: 10.

[5]. Podolsky, V.E. and S.S. Tolstykh, 2006. Improving the Efficiency of Regional Educational Computer Networks Using the Elements of Structural Analysis and Complexity Theory. Moscow: Mashinostroenie.

[6]. Tikhonov, A.N., S.V. Mishchenko, V.E. Podolsky and S.S. Tolstykh, 2004. Features of Mathematical Modeling of Modern Computer Networks in Education (Vol. 1). St. Petersburg, pp: 78-79.

[7]. Tolstykh, S.S. *et al*, 2009. Constructing of the Optimal Block-Concurrent Calculation Modules in a Distributed Computing Environment. In New Information Technology and Quality Management: Proceedings of the International Symposium, Turkey, pp: 160-162.

[8]. Fedorov, R.V., Yu.V. Osetrov and S.S. Tolstykh, 2009. Designing the Architecture of a Scientific Information System to Support the Study of Structural Complexity. In V.P. Gergel (Ed.), Microsoft Technologies in the Theory and Practice of Programming. Conference Materials, Nizhny Novgorod: Publishing House of the Nizhny Novgorod State University.

[9]. Mainika, E., 1981. Optimization Algorithms on Networks and Graphs. Moscow: Mir.

[10]. Lipaev, V.V., 2001. Providing the Software Quality. Methods and Standards. Moscow: SINTEG.

[11]. Klir, G., 1990. Systemology. Automation of the Solutions of System Problems. Moscow: Radio i Svyaz.

[12]. Casti, J.L., 1982. Large Systems. Connectivity, Complexity and Catastrophe. Moscow: Mir.

[13]. Savage, J.E., 1998. Computational Complexity. Moscow: Factorial.

[14]. Razborov, A.A. About Computational Complexity. Mathematical Education, 3: 127-141.

[15]. Kuzyurin, N.N. and S.A. Fomin, 2007. Efficient Algorithms and Computational Complexity. Moscow: MIPT.

[16]. Lazdin, A.V. and O.F. Nemolochnov. Complexity Estimate for the Graph of a Functional Program. Scientific and Technical Bulletin of SPbGITMO (TU), 6: 112-117.

[17]. Voevodin, V.V. and Vl.V. Voevodin, 2002. Parallel Computing. St. Petersburg: BHV-Petersburg.

[18]. Tarasov, A.G., 2009. Expandable Monitoring System for a Computing Cluster. Computational Methods and Programming, 10: 147-158.

[19]. Kutepov, V.P., D.V. Kotlyarov, V.N. Malanin and N.A. Pankov, 2007. Environment of the Object-Oriented Graph-Scheme Flow Parallel Programming for Multi-Core Clusters. In Proceedings of the Sixth International Scientific and Practical Seminar "High-Performance Parallel Computing on Cluster Systems", St. Petersburg, 12-17 December

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2006 (Vol. 1), St. Petersburg: St. Petersburg State University, pp: 253-258.

[20]. Nicolis, G. and I. Prigogine, 1990. Exploring Complexity. Moscow: Mir.

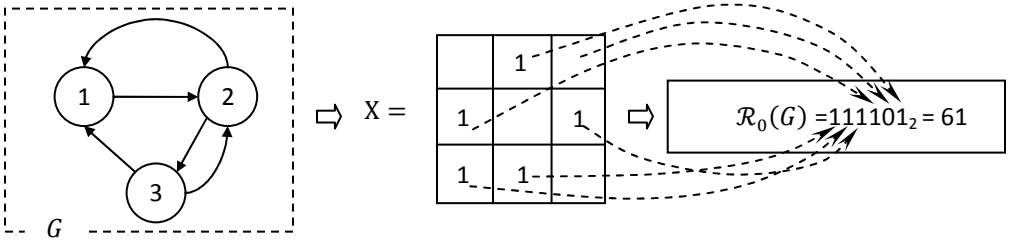
[21]. Asharina, I.V., 2009. Identifying Complexes in the Clustered Computing Systems. In Proceedings of the Ninth International Conference-Workshop "High- Performance Parallel Computing on Cluster Systems", Vladimir.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Remarks

1. In this case, a specification is appropriate: a tree of vertices.
2. An alternative is the load on the vertices in the form of a vertex potential. In our opinion, such an approach may greatly complicate the evaluation of complexity, especially when the digraph is strongly connected. Under such approach, any calculations associated with the estimates of complexity presuppose solving the Kirchhoff system of equations.
3. The issues of completeness of the digraph selection are not considered here, it is a separate research topic.
4. In order not to confuse the contour SCSs with bicomponents, we will use as superscript not "~", but the sign "."

APPENDIX



$$\mathcal{R}_0(G) = 61 + 1984 = 2045$$

$$\mathcal{R}_0(\hat{G}) = 63, k = \lfloor \log_2 63 + 1/2 \rfloor = 6 \Rightarrow \mathcal{R}_1(G) = 2^6 + 2^7 + 2^8 + 2^9 + 2^{10} = 1984$$

Figure 4. An example of a digraph invariant

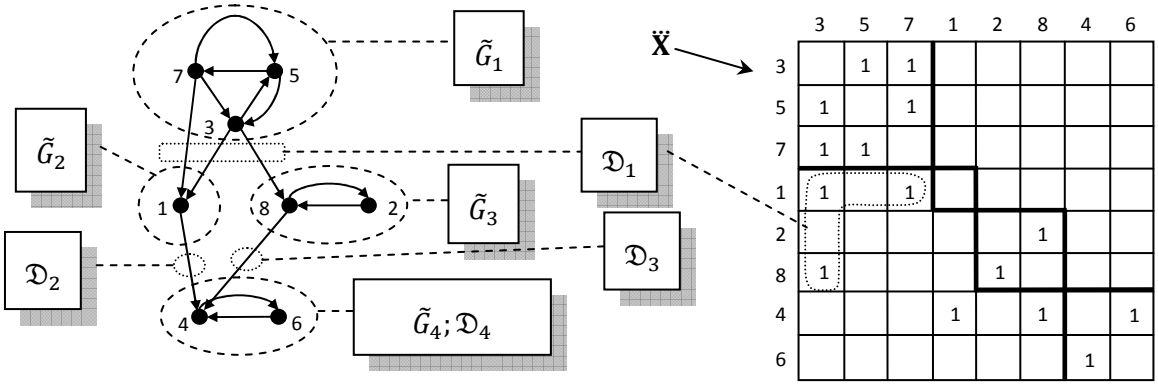


Figure 7. Illustration to the condition (16)

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

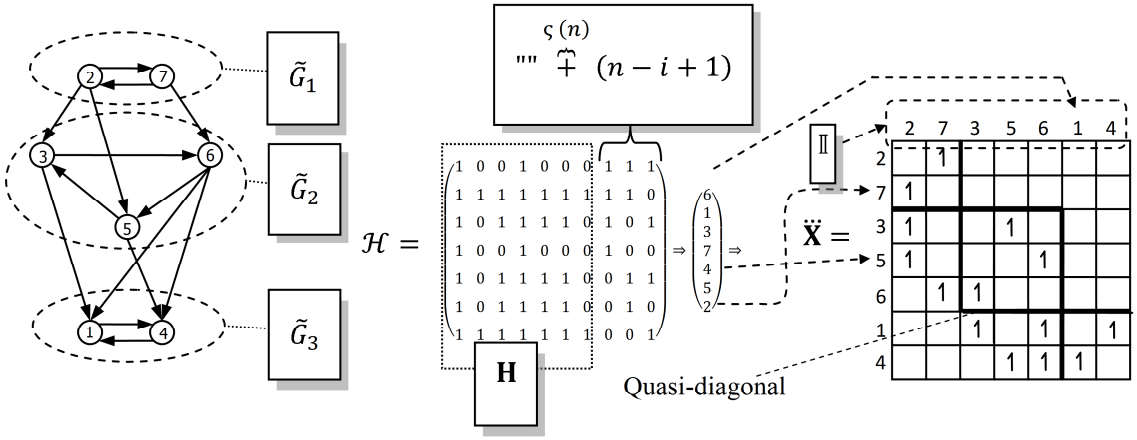


Figure 8. Illustration to anti-lexicographic sorting of the vector $\mathcal{H}$

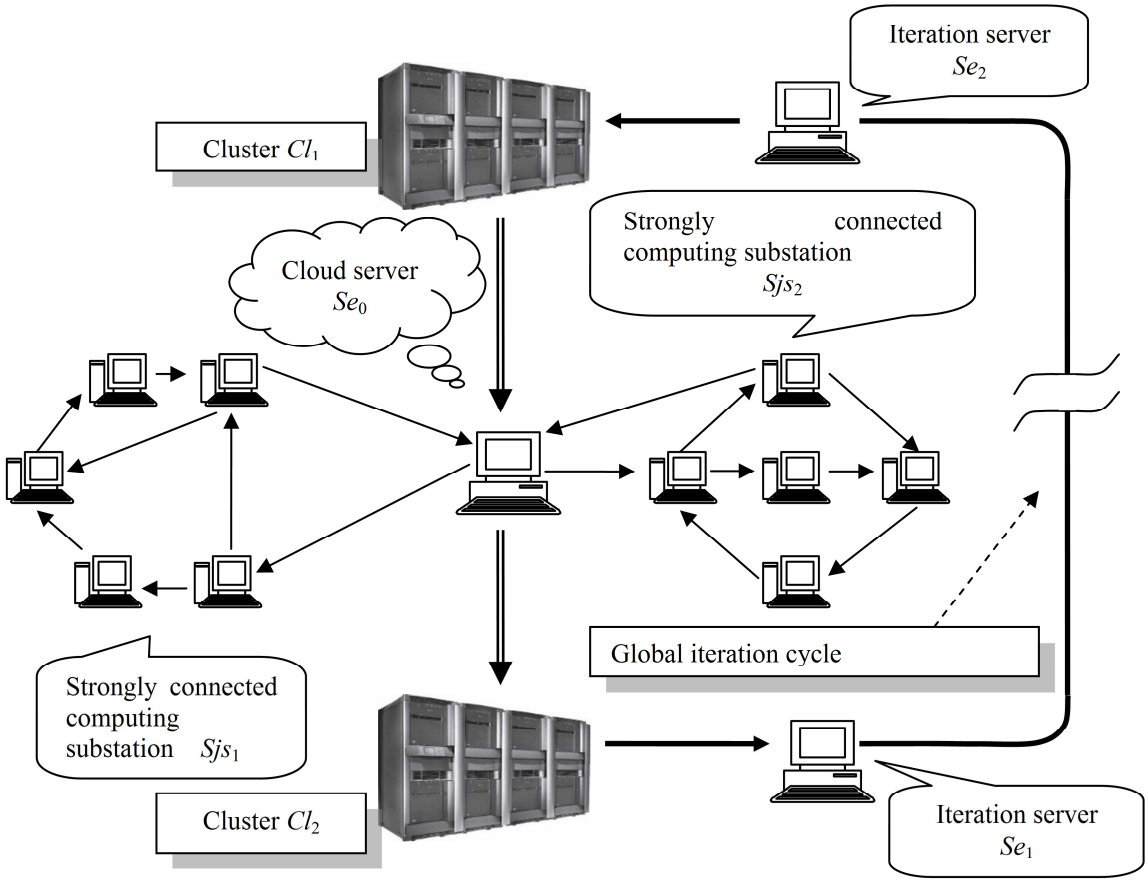


Figure 10. Example of a grid system with global iterations and a cloud server

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

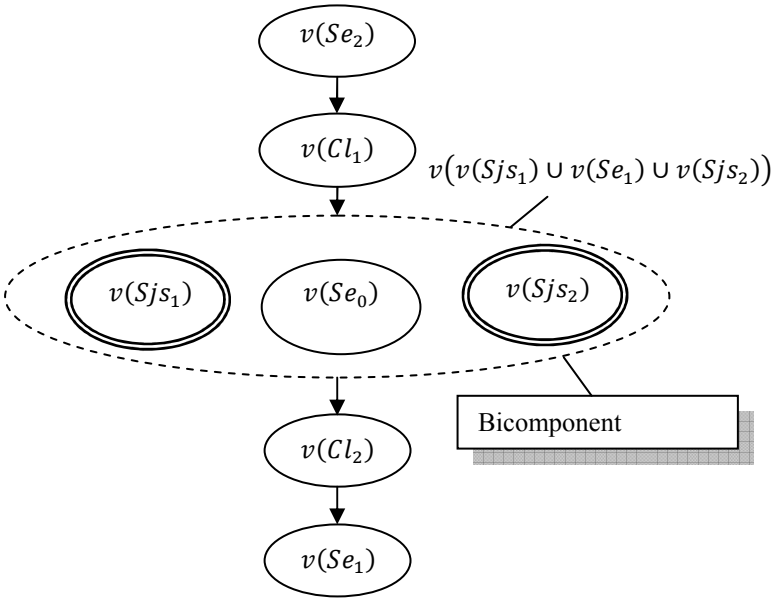


Figure 11. Structuring of a grid system: the case of one bicomponent

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

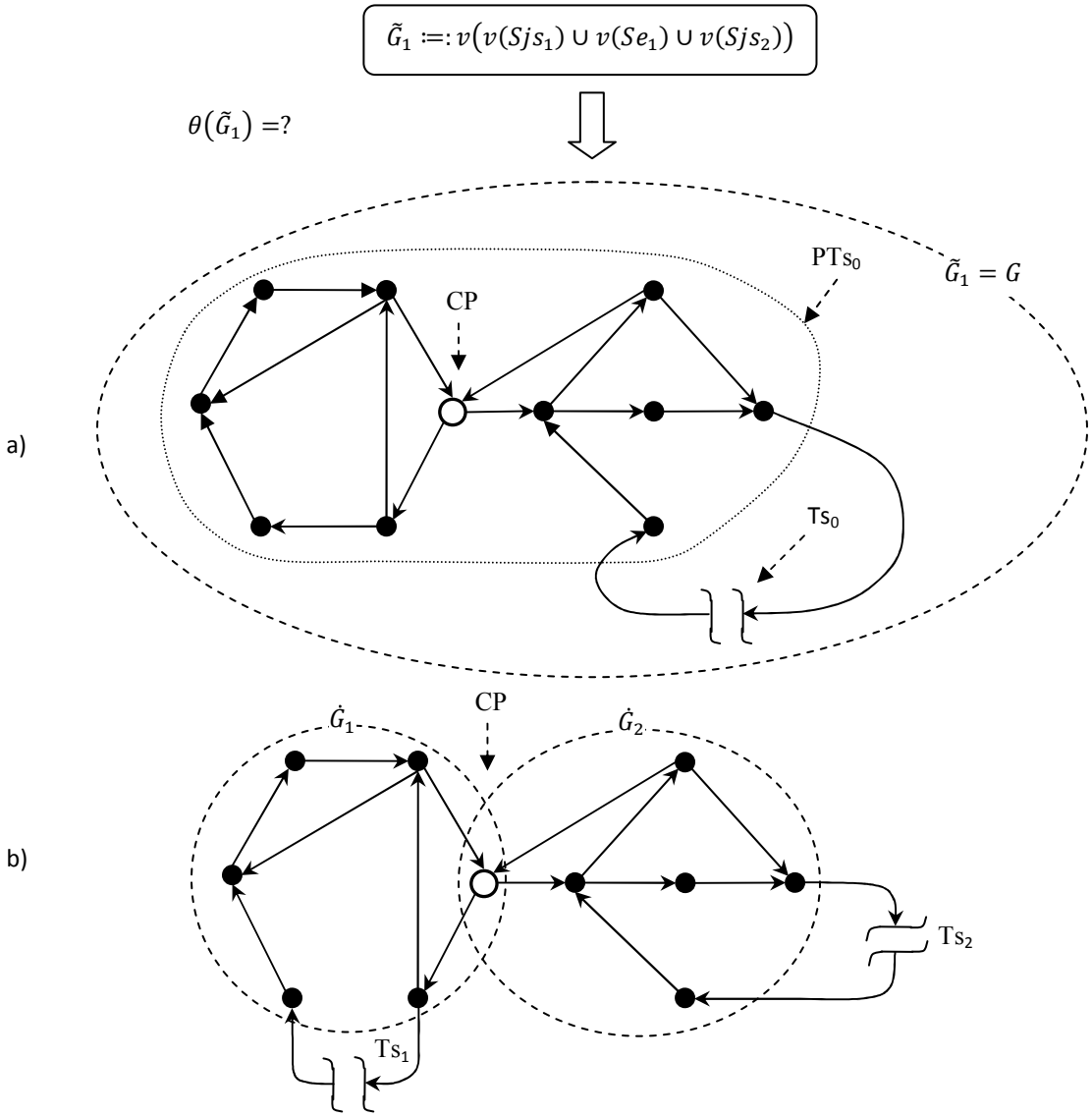


Figure 12. Variants for decomposition of the CCS digraph in Figure 11: a) without taking into account a cutpoint; b) taking into account a cutpoint

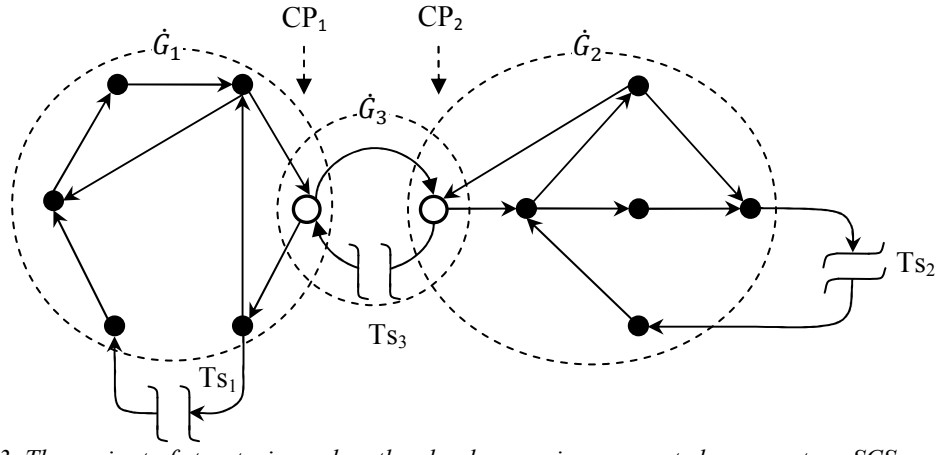


Figure 13. The variant of structuring, when the cloud server is represented as a contour SCS

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

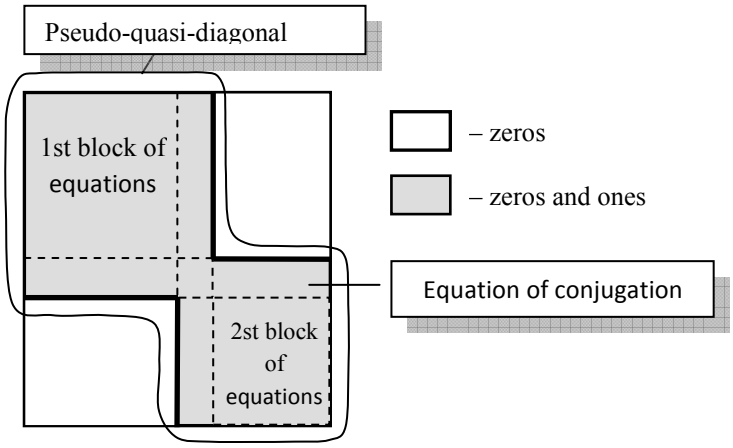


Figure 14. General form of the adjacency matrix for a system of equations of large dimension with the equation of conjugation of blocks

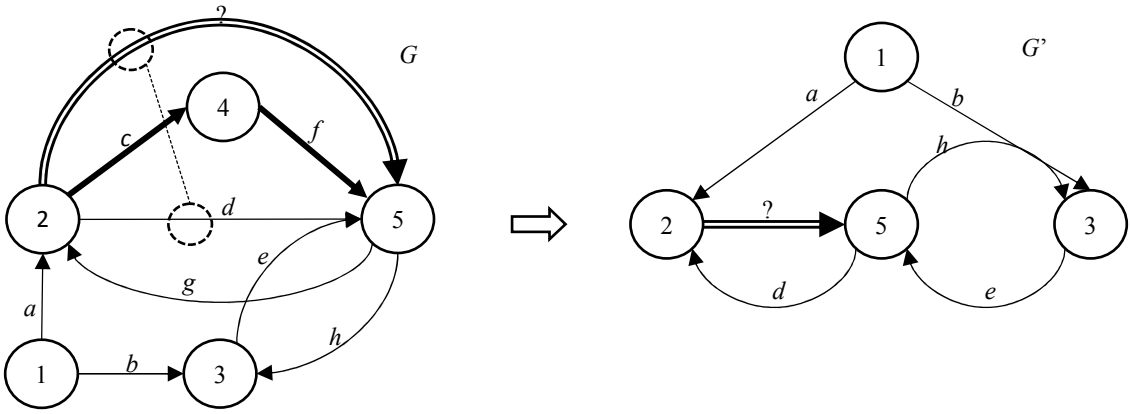


Figure 15. Example of a CCS digraph with redundant elements

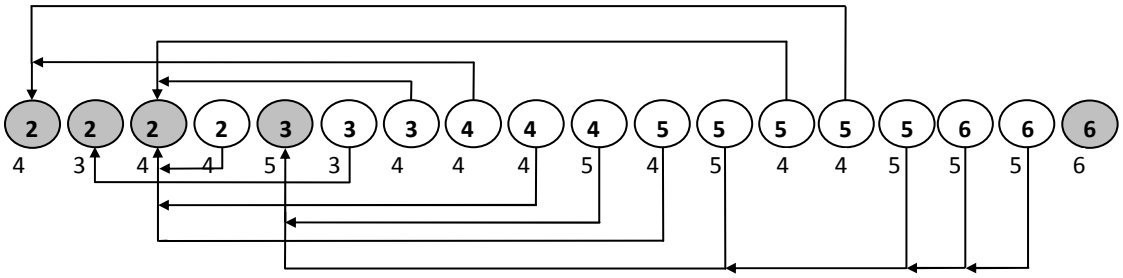


Figure 17. The codes $\mathcal{R}_0(G)$ of digraphs with three vertices and the number of arcs (below)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

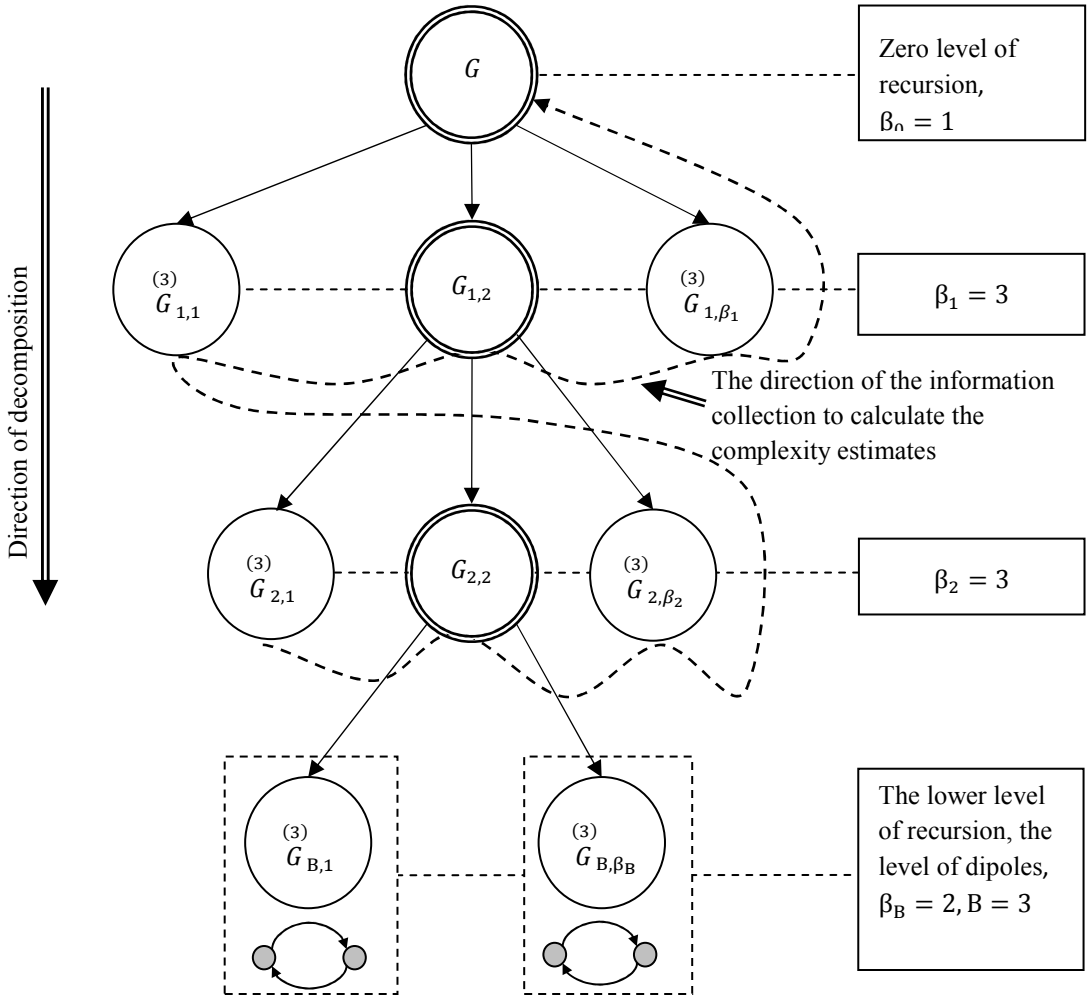


Figure 19. An example of tree recursion: G is the CCS digraph in its original state; $B = 3$ is the number of levels of hierarchy; $\beta_i, i = 0..B$ is the width of the levels in the tree of recursion

Journal of Theoretical and Applied Information Technology
10th December 2015. Vol.82. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

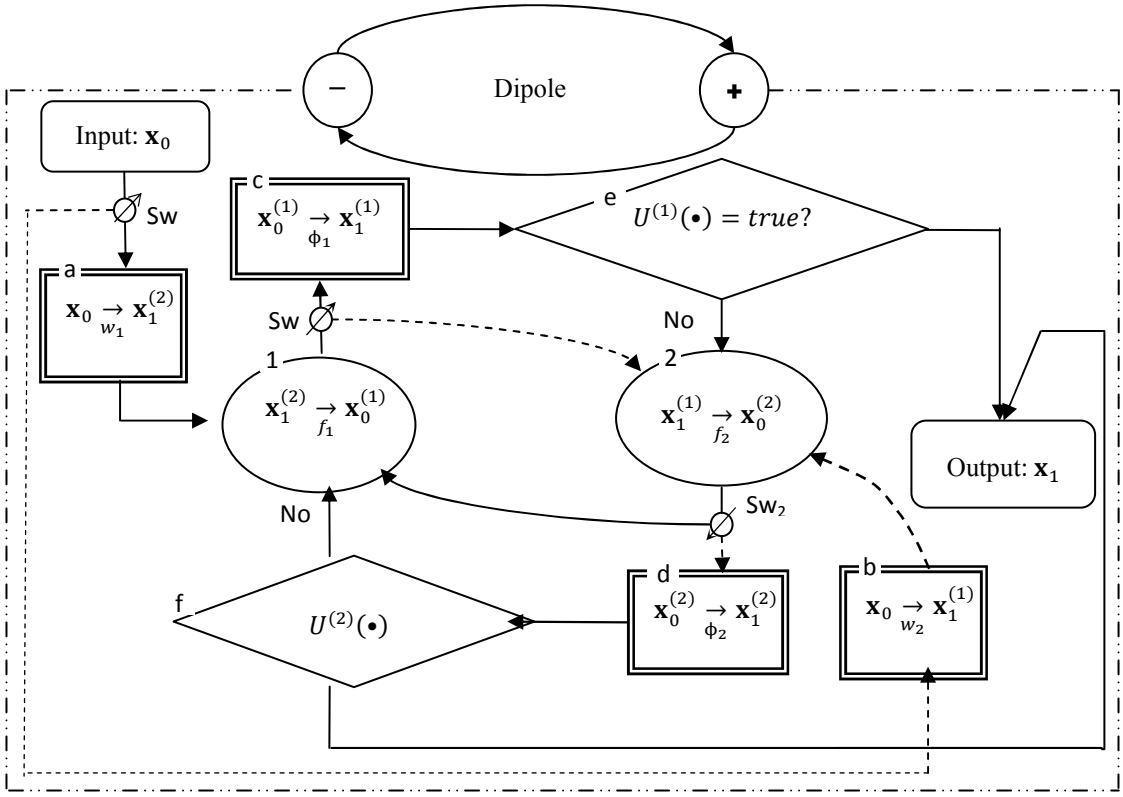


Figure 20. Dipole as a procedural model of iterative CCS