

# Interactive 3D Representation as a Method of Investigating Information Graph Features

Alexander Antonov ✉ and Nikita Volkov

Lomonosov Moscow State University, Moscow, Russia  
asa@parallel.ru, volkovnikita94@gmail.com

**Abstract.** An algorithm information graph is a structure of wide variety. It can tell a lot about algorithm features, such as computational complexity and resource of parallelism, as well as about sequential operations blocks within an algorithm. Graphs of different algorithms often share similar regular structures — their presence is an indicator of potentially similar algorithm behavior. Convenient, interactive 3D representation of an information graph is a decent method of researching it; it can demonstrate algorithm characteristics listed above and its structural features. In this article we investigate an approach to creating such representations, implement it using our AlgoView system and give examples of using a resulting tool.

**Keywords:** Information graph · Parallelism · AlgoWiki · AlgoView · level parallel form.

## Introduction

An information graph of an algorithm is a DAG (directed acyclic graph), where vertices stand for operations within an algorithm and edges stand for data dependencies [1]. Its size and structure can thus depend not only on particular algorithm features, but also on input data size and on an exact execution sample (the latter happens only in case of a nondeterministic algorithm). Information graph can demonstrate algorithm's resource of parallelism, determine primary data flows within it and show parts that require most computation to be executed. Information graphs of different algorithms can possess similar regular structures, demonstrating possible similarities in algorithms' execution processes.

The graph itself can be represented in many forms, which allow to highlight some particular features of its structure. These representations include the level parallel form [1] and projections of a multidimensional graph on a hyperplane with less dimensions. The number of dimensions is an algorithm characteristics which can be defined as the maximum depth of cycle nests within an algorithm. If it's 3 or less, one can easily represent a graph in 3-dimensional space without hiding its regular structure. Otherwise more complex methods may be required to display such a graph with all regularities preserved. These include graph projections and macrographs (a generalization of information graphs, where vertices can stand not only for operations, but also for whole information graphs of algorithm's subtasks).

2 A. Antonov, N. Volkov

## 1 Graph Visualization

Representing a graph in a visual form is overall a complex task with many approaches to its solution. The exact approach depends on the primary goal of a given graph visualization task. Most well-developed and widespread techniques aim for automatically built images of really large graphs of, say, social networks or worldwide web parts, with related vertices being grouped into clusters. An example of this approach can be found in [2]. We, however, deal with small graphs that have a feature their bigger brothers don't — a strict, regular structure. Information graphs therefore have much more in common with data flow and control flow graphs used in compilation theory [3]. The limited size of studied graphs comes from their definition — we can take input data size as small as we want, if the graph still retains its regular structure.

## 2 The AlgoWiki Project

We came up with an idea of the AlgoWiki project a few years ago [4, 5]. It's main purpose is to describe algorithms accenting their resource of parallelism, possible implementations for massively parallel systems, researching scalability [7] and efficiency. However, new possible problems to solve appeared during it's development. These include investigating less-known algorithm characteristics, such as data locality [8], building up a classification system for tasks, methods and algorithms, searching for algorithms that could be used as benchmarks and creating supercomputer ratings based on these benchmarks. AlgoWiki is an open web-resource [6] powered by MediaWiki engine. This project is supported by Russian Science Foundation and officially led by a well-known scientist, professor J. Dongarra (University of Tennessee, USA).

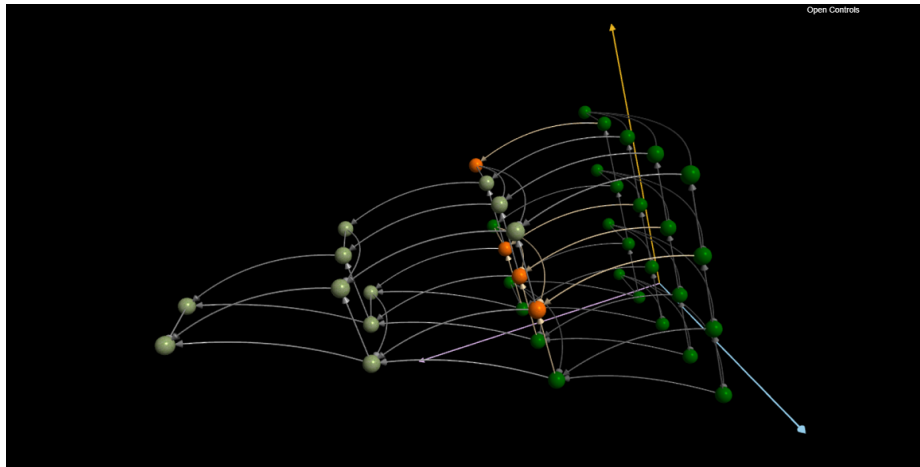
## 3 An Interactive Representation — Features and Problems

A visual representation of an algorithm graph is definitely a powerful instrument for researching its structure. There are, however, so many different information graphs, that a viable and logical form of visual representation simply can't be the same for all of them. Modifications of a visual representation are very useful for displaying certain algorithm features.

Moreover, our AlgoView system used in AlgoWiki to provide visual representations of information graphs [9] generates 3D representations automatically, based on XML algorithms descriptions (which, in their turn, are based on actual program code). This system defines every vertex of an information graph and its position within a generated representation. Edges are defined by pairs of vertices they connect. This information doesn't give a unique representation of vertices and edges, since we don't know, how exactly graph edges should be displayed. This causes certain technical difficulties.

### 3.1 Level Parallel Form of an Information Graph

A level parallel form is one of many ways to show an information graph. All graph vertices are placed on a certain level, which is defined by two simple rules. A vertex without data dependencies is placed on level one. For any other vertex all sources of data dependencies are assigned to a certain level first; the vertex is then given a level of maximum source level plus one. The main purpose of such a representation is to show algorithm's resource of parallelism. Another one (used in our interactive representations) is to follow operation executions within an algorithm. Our method of displaying an algorithm level parallel form is shown in Figure 1. The tool we created to show information graphs is available online and interactive, so users should have a certain portion of control over what they see on their screens. In the upper right corner of Figure 1 a minimized control panel used to manipulate the visualization and provide information can be seen. It consists of several menus which can be maximized separately and allow to easily acquire visualization projections, highlight level-parallel form levels, vertex types, display information like axis names etc., but don't possess scientific value themselves and thus won't be referred to later on.



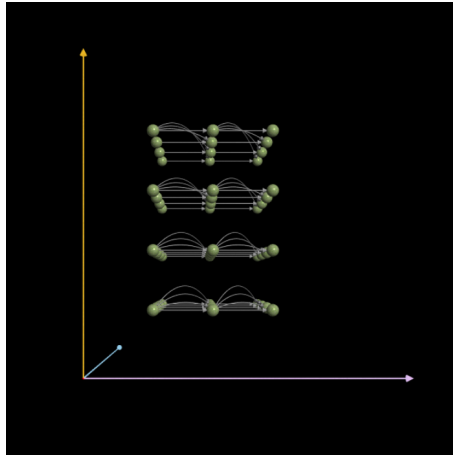
**Fig. 1.** Possible way to display a level parallel form; vertices from current level are shown in orange, previous levels are dark green, upcoming levels are light green.

### 3.2 Information Graph Projections

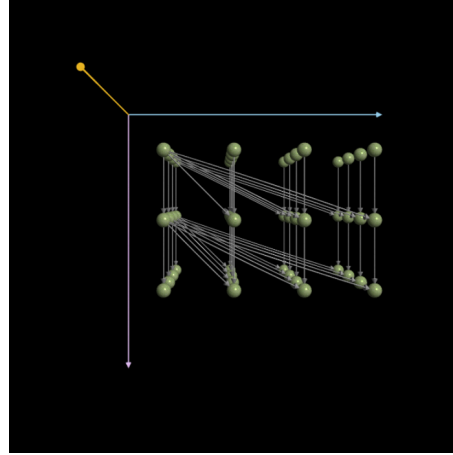
As it was said before, our visual representation of an information graph is 3-dimensional. However, regular structures within a graph aren't necessarily 3-dimensional themselves. They can have both more and less dimensions. An example of a "1-dimensional" structure is data broadcast. A 3-dimensional representation can be projected on axis or plane to better display such low-dimensional

4 A. Antonov, N. Volkov

structures. A good projection brings more clearness into information graph structure. For example, in Figure 2 data dependencies are almost hidden, while in Figure 3 one can clearly see a basic data broadcast.



**Fig. 2.** A projection of an information graph on oYZ plane; broadcast is almost hidden.



**Fig. 3.** A projection of an information graph on oXZ plane; broadcast can be clearly seen.

### 3.3 Removing Edges Intersections

Intersections of edges within an information graph visual representation can be divided into several categories: self-intersections, edges intersecting each other, edges passing through vertices that are neither edge beginning nor edge ends. Currently we use only one method to avoid too many edge intersections. If distance between the start and end vertices of an edge surpasses a certain threshold, then Bezier curves are used to layout edges in 3-dimensional space instead of straight lines. Curving extent can also be set to depend on distance between vertices. A good display of this idea is given in Figure 4.

Sometimes this algorithm doesn't allow to remove all edge intersections, as shown in Figure 5. However, it at least excludes the possibility of edges completely imposing each other. In some cases intersection practically don't affect one's ability to distinct different edges. An example of such a graph will be shown later in Figure 11.

### 3.4 Outer Parameters and Point of View

Besides key questions described above, one needs to address lesser tasks when creating a visual representation of an information graph. The most obvious of

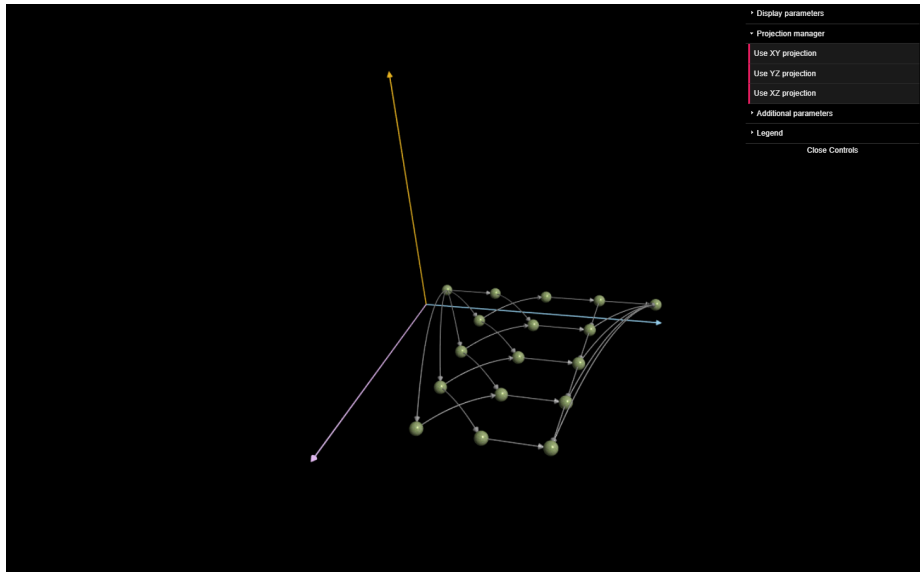


Fig. 4. Broadcasting is displayed with the help of Bezier curves.

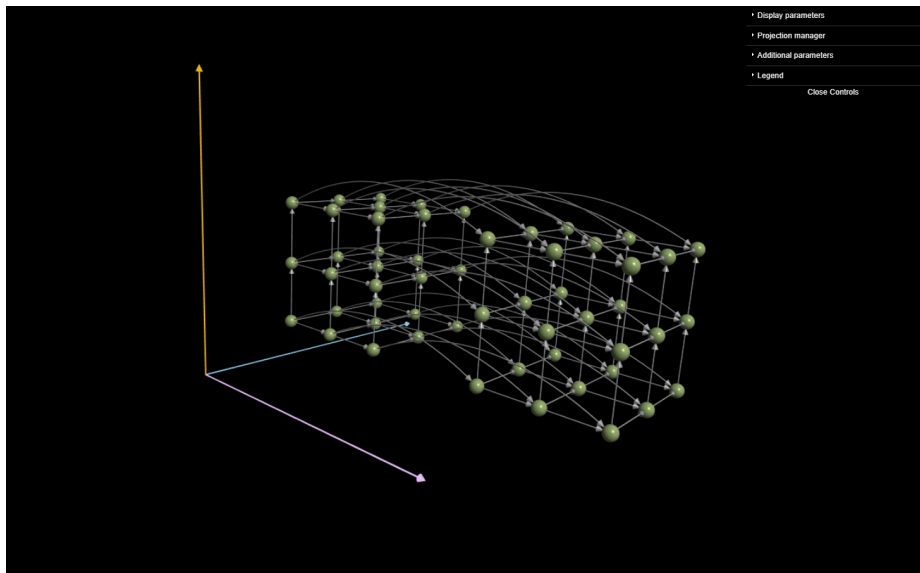


Fig. 5. Edges still intersect, but don't imposition each other.

6 A. Antonov, N. Volkov

them is choosing a correct point of view for a given representation. With point of view being a generalization of graph projections, different points of view can simplify or complicate one's understanding of an algorithm structure.

We decided to allow users to choose points of view themselves, while certain points of view (such as projections) are saved, so current point of view can always be reset to one of those.

Information graphs are also different for every given size of input data, which is controlled by so-called outer parameters. These affect not only the size of regular structures within an information graph, but also define whether some vertices and edges are present in a graph or not. A good example of that is Householder algorithm for QR-decomposition of a matrix [10]. There are  $NI$  additional blocks in it, and data is broadcasted from the first block to all of these during every step of an algorithm. If  $NI = 1$ , however, there's only one block and the broadcast nature of its data dependency from the first block is hidden.

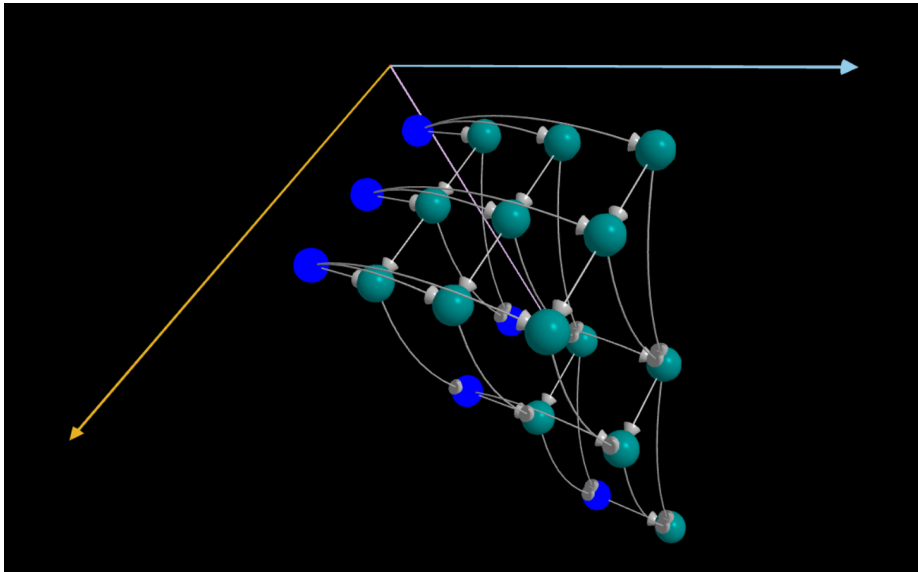
### 3.5 Macrographs

In information graph definition nothing is said about what operations displayed by vertices are, but it's usually assumed that they are atomic tasks that cannot be further subdivided into smaller parts. macrograph is a generalization of an information graph, in which operations displayed by vertices can be pretty much anything, including whole information graphs of an algorithm subtask. Two macrograph vertices have a data dependency between them, if there is a data dependency between any pair of information graph vertices they represent. A concept of macrograph is really useful for representing algorithms and methods that basically consist of several more simple algorithms. Here we give a simple display of a macrograph concept. In Figure 6, you can see the already familiar Givens method from Figure 1, except for now vertices as displayed as macros, not just simple operations. In Figure 7, there's a more thorough structure view of a cyan colored macro vertex from Figure 6, where green vertices stand for normal operations and red octahedrons stay for input and output data.

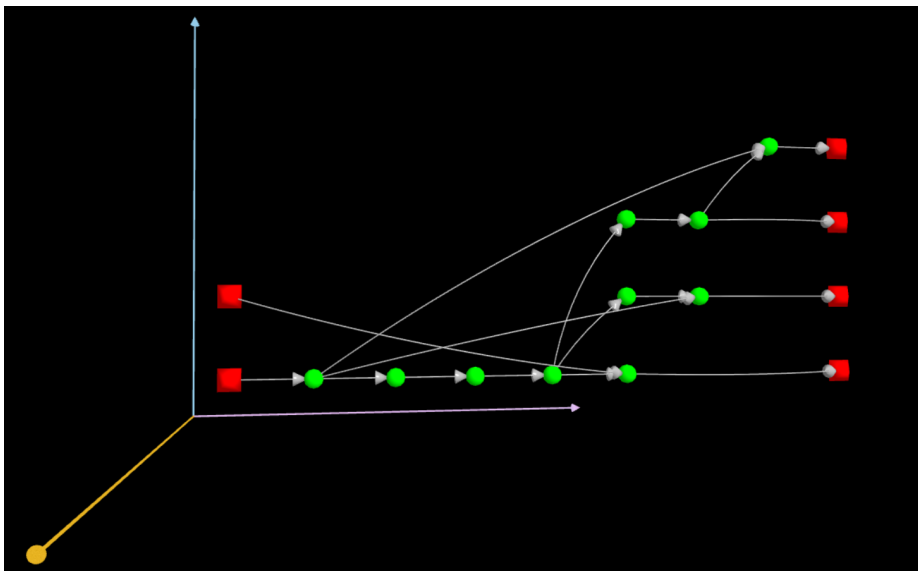
## 4 A Library of Typical Information Graph Structures

### 4.1 The AlgoView System

Our AlgoView system, described in detail in [9], is a set of applications and scripts used for creating interactive 3-dimensional representations of information graphs. Here we give only a brief overview of it. The AlgoView system includes two main functional tools — a 3D model generator written in C++ and a viewer, which is based on a set of web-pages with JavaScript and WebGL utilities attached to it. It's supported by server-executed node.js code, which brings more functionality to an interactive representation, for example, allowing user to re-generate 3D-models on server if needed.



**Fig. 6.** Givens QR-decomposition method for  $N = 3$ . At each plane, all vertices but the broadcasting ones display a 2D-vector rotation operation.



**Fig. 7.** A more detailed structure of a 2D vector rotation operation from Figure 6.

8 A. Antonov, N. Volkov

**Generator.** An input for the generator is an XML file of known format. XML files are also built automatically, but XML file builder doesn't belong to the AlgoView system. A set of 3D models representing the information graph is generated. User is asked to provide values for all outer parameters listed in an XML file.

**Viewer.** Users are supposed to interact with the viewer as follows: prepared 3D models are stored on server. Users open web-pages containing 3D representations in their browsers and JavaScript code is executed on client side. This code is responsible for showing models in browser window and manipulating them.

**Server part.** This includes node.js code executed on server, which allows to upload new models to same web-page and to generate 3D models with new parameters on server (where a copy of generator application is stored).

One can read more about AlgoView in the link given above. However, server part isn't described there, since it wasn't even planned at that time. It's being developed at the moment.

#### 4.2 A Library of Typical Information Graph Structures Based on AlgoView

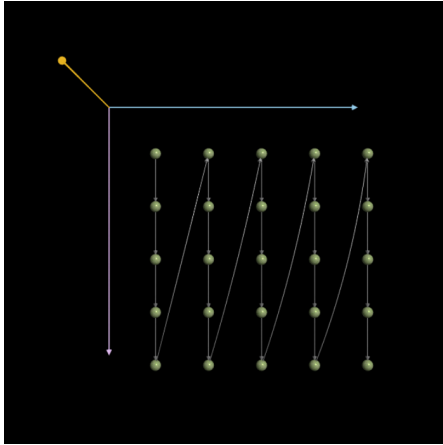
The AlgoView system provides quite a variety for possibilities to display information graphs. These representations are already used to further describe certain algorithms in AlgoWiki [11], but they can also be used for another purposes. Many different algorithms possess similar regular structures. Some of these regular structures are very well known in terms of how deeply their resource of parallelism and other characteristics are investigated. They are also encountered in various algorithms very often and can be called typical. We decided to use the AlgoView system to create a library of such typical regular structures.

The most common ones are 2- and 3-dimensional groups of vertices, which form up a grid or a group of grids. The main difference between these structures lies in the way of how data dependencies are organized. It's convenient to divide dependency patterns into categories not only by addressing them the way its done in level parallel form, but also by paying attention how much coordinates in n-dimensional space are different in a pair of vertices connected by a graph edge. These categories are good for describing a complicated dependency pattern as a combination of more simple ones. Moreover, they correspond very well to a formal description of an information graph given in XML files, which are used by the AlgoView system. Some of these typical structures are:

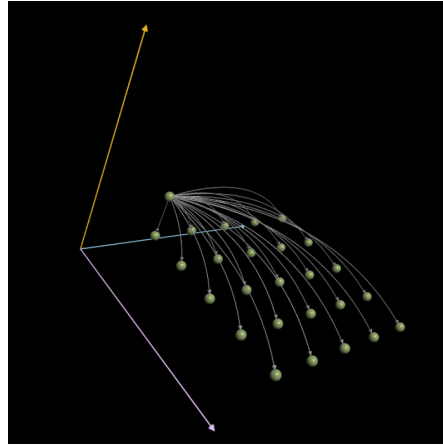
- N-dimensional sequential structures, as shown in Figure 8;
- N-dimensional broadcasts, as shown in Figure 9;
- Independent sets of sequential structures or broadcasts (Figure 10);
- Same sets of sequential structures and broadcasts, but with dependencies between them (Figure 11);



- Schemes with oblique parallelism [10], as shown in Figures 12 and 13;
- Same as previous category, but with additional broadcasts.



**Fig. 8.** A 2-dimensional sequential structure.



**Fig. 9.** A 2-dimensional broadcast.

This basic set is already enough to display many structures seen in widely used algorithms, such as [11] and [12]. More complex structures can be described as a combination of these simple ones. Examples are in Figures 14 and 15.

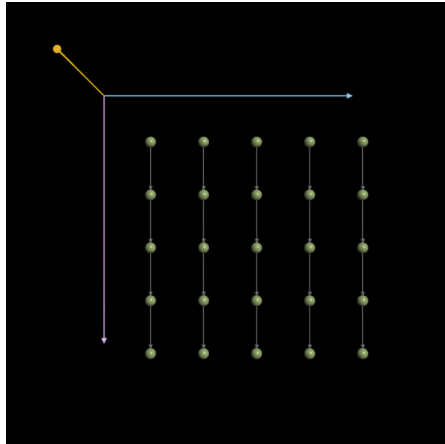
In addition to its main purpose, this library can, from our point of view, help in solving other tasks. One of them is testing and further developing the AlgoView system. In case of complex information graphs structures the system build correct, but yet non-aesthetic 3D-models. Since information graph complexity doesn't have any limits (except for the maximum number of edges for a graph of given size), we aren't aiming for an ideal 3D-model generation system. However, paying attention to structures common in real algorithms is, in our opinion, definitely worth it.

Our plan is also make this regular structures library public with same intentions as the whole AlgoWiki project. Just like with algorithms, we plan to further classify these structures and create description pages for all of them. Every page will contain an interactive representation of a given structure and examples of algorithms utilizing it.

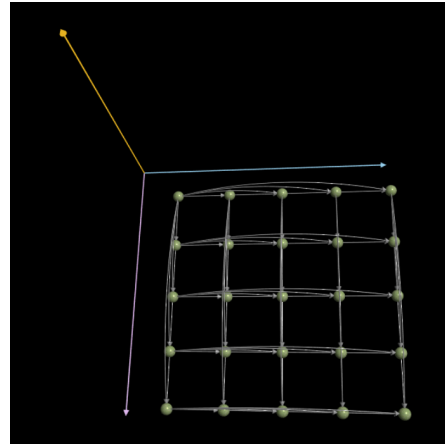
## 5 Conclusion

In this paper, we have given a review of various algorithm features that can be represented within an information graph, as well as methods to do so. We have also provided data on our approach to visualizing information graphs considering

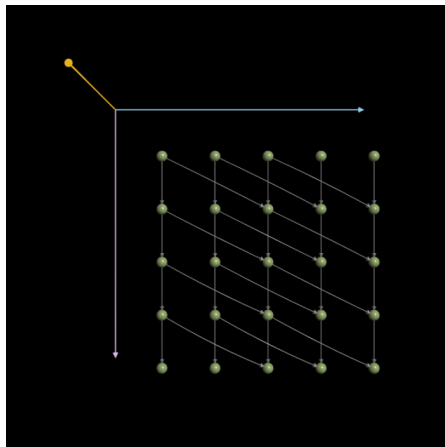
10 A. Antonov, N. Volkov



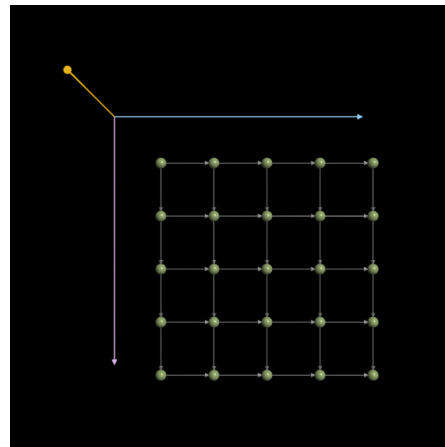
**Fig. 10.** A set of 1-dimensional, independent, sequential structures.



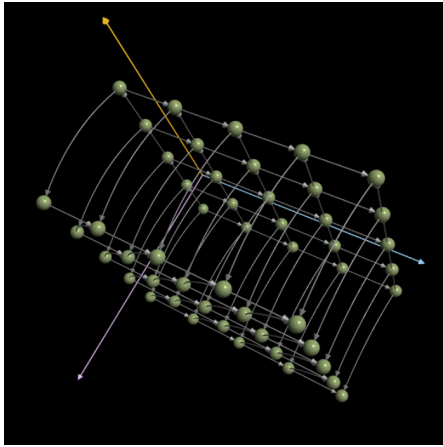
**Fig. 11.** Two sets of 1-dimensional broadcasts, dependent on each other.



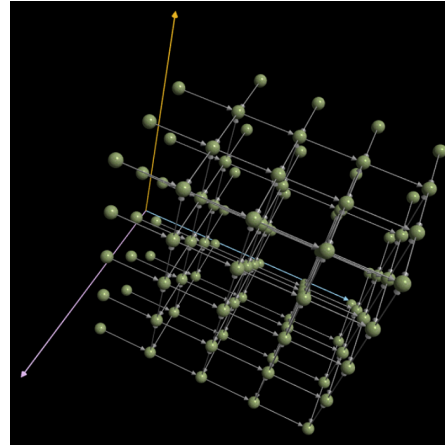
**Fig. 12.** A parallel scheme.



**Fig. 13.** A scheme with oblique parallelism.



**Fig. 14.** Two 2-dimensional vertex groups with parallelism, connected to each other.



**Fig. 15.** A 3-dimensional vertex group, being a set of 2-dimensional schemes with oblique parallelism

these features and provided an updated brief description of our AlgoView system. In addition, our two latest ideas were discussed in this paper. The first one is about creating a public library of typical structures found in information graphs with the help of interactive 3D representations. A list of such structures was described and examples of 3D representations were provided. The second idea is all about finding algorithms that could be good benchmarks for massively parallel systems, since we think a good sign of such an algorithm would be numerous typical structures within it. Further research and implementation of these two concepts is therefore our primary concern.

## 6 Acknowledgements

The results described in sections 1–2, 4 were obtained in Lomonosov Moscow State University with the financial support of the Russian Science Foundation (Agreement № 14–11–00190). The research is carried out using the equipment of the shared research facilities of HPC computing resources at Lomonosov Moscow State University supported by the project RFMEFI62117X0011.

## References

1. Voevodin, V., Voevodin, Vl.: Parallel Computing. BHV-Petersburg, St. Petersburg (2002)
2. Hu, Yi., Shi, L.: Visualizing large graphs. WIREs Computational Statistics. Vol. 7. Issue 2. Pp 115-136 (2015) DOI: 10.1002/wics.1343
3. Gold, R.: Control flow graphs and code coverage. Applied Mathematics and Computer Science. Vol. 20. No. 4. Pp. 739–749 (2010) DOI: 10.2478/v10006-010-0056-9

12 A. Antonov, N. Volkov

4. Voevodin, V., Antonov, A., Dongarra, J.: AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. *Supercomputing Frontiers and Innovations*. Vol. 2. No. 1. Pp. 4–18 (2015) DOI: 10.14529/jsfi150101
5. Voevodin, V.I., Antonov, A., Dongarra, J.: Why is it hard to describe properties of algorithms? *Procedia Computer Science*. Vol. 101 (2016). Pp. 4–7. DOI: 10.1016/j.procs.2016.11.002
6. Open Encyclopedia of Parallel Algorithmic Features, <http://algowiki-project.org/en>. Last accessed 13 Apr 2018
7. Antonov, A., Teplov, A.: Generalized Approach to Scalability Analysis of Parallel Applications. *Lecture Notes in Computer Science*. Vol. 10049. Pp. 291–304 (2016) DOI: 10.1007/978-3-319-49956-7\_23
8. Alexander Antonov, Vadim Voevodin, Vladimir Voevodin, Alexey Teplov. A Study of the Dynamic Characteristics of Software Implementation as an Essential Part for a Universal Description of Algorithm Properties. 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing Proceedings, 17th–19th February 2016. Pp. 359–363. DOI: 10.1109/PDP.2016.24
9. Antonov, A.S., Volkov, N.I.: An AlgoView Web-visualization System for the AlgoWiki Project. *Communications in Computer and Information Science*. Springer International Publishing AG. Vol. 753. Pp. 3–13 (2017) DOI: 10.1007/978-3-319-67035-5\_1
10. Householder (reflections) method for the QR decomposition of a square matrix, real point-wise version, [http://algowiki-project.org/en/Householder\\_\(reflections\)\\_method\\_for\\_the\\_QR\\_decomposition\\_of\\_a\\_square\\_matrix,\\_real\\_point-wise\\_version](http://algowiki-project.org/en/Householder_(reflections)_method_for_the_QR_decomposition_of_a_square_matrix,_real_point-wise_version). Last accessed 13 Apr 2018
11. Givens method [http://algowiki-project.org/en/Givens\\_method](http://algowiki-project.org/en/Givens_method). Last accessed 13 Apr 2018
12. Dense matrix multiplication (serial version for real matrices) [http://algowiki-project.org/en/Dense\\_matrix\\_multiplication\\_\(serial\\_version\\_for\\_real\\_matrices\)](http://algowiki-project.org/en/Dense_matrix_multiplication_(serial_version_for_real_matrices)). Last accessed 13 Apr 2018