Bull. Nov. Comp. Center, Comp. Science, 9 (1998), 59–66
© 1998 NCC Publisher

TOPAS: Java-based system for investigation of parallel algorithms mapping*

O.G. Monakhov

TOPAS (Test and Optimization of Parallel Algorithms and Structures), a programming tool for visualization, animation and investigation of sequential and parallel algorithms for mapping of parallel program graphs into graph structure of parallel computer systems is presented. The tool is implemented in Java and is accessed on WWW: `http://rav.sscc.ru/~monakhov/topas.html`.

Introduction

Methods for automatic mapping of a graph structure into another graph structure are widespread, a lot of mapping algorithms is built up and checked. These methods can be used for the mapping of modules of parallel programs into processors of parallel computing systems, as well as for special data decompositions and solving the problems of operations research, graph theory, combinatorial optimizations, networking, etc. In other words, mapping algorithms are important for various applications. So, to acquire the knowledge about these algorithms we are developing the system TOPAS.

A system presented in this paper allows the users in their graph research to be concentrated on work with algorithms and models, leaving apart difficulties, related to creating and saving entrance data, programming in conventional languages, etc. The system supports visualization of graphs, mapping processes, as well as visualization of mapping results. In addition, it can be used to define some "atomic" operations (for example, partitioning of the complex data structures, e.g., irregular meshes) for more complex problems being specified within the framework of the VIM technology [1]. The system has been written in Java as an applet, which allows to place it on HTML-pages (`http://rav.sscc.ru/~monakhov/topas.html`), to interact and to demonstrate algorithms at any point of the world where Internet is accessible (this approach is analogous to [2]). We call this system by TOPAS. In general, it is related to the development of a problem-solving environment which integrates multimedia, parallel and distributed Internet

*Supported by the Russian Foundation for Basic Research under Grants 96-01-01632, 99-07-90422.

technologies for specification (programming) of application algorithms, and the control of mapping strategies and data visualization.

Different parts of the system (the window editor of graphs, modules for imaging the entrance data, visualization and animation of mapping algorithms, representing and saving the results, etc.) can also be used independently.

1. TOPAS Structure

The system consists of the following parts:

Core. The TOPAS's core is an applet that displays initial data, calls the graph editor (for data editing) and starts necessary algorithms.

Graph's icons. They are used by the core to display the initial data and are small images of the graphs.

Graph editor. This editor allows to safe and restore structures of weighted graphs, to modify them and insert structures prepared, such as lattices, rings, hypercubes, etc.

Supervising module. This module displays intermediate results of algorithms' work in a convenient form and allows to stop the work and to study intermediate results in more detail.

Module for representing and saving results. This module allows to represent results of the algorithm's work in an obvious form, and also to save them on disk.

Libraries of sequental and parallel mapping algorithms.

Help module. This module displays information about system, interfaces, problem and mapping algorithms.

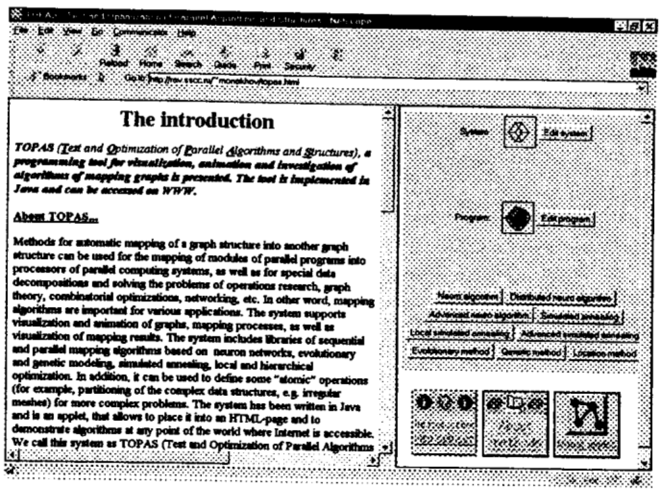
Internal representation of graph structure. This format allows to connect all components of the system.

Programmer's interface. It allows to insert easily new algorithms into the system.

Distributed agent system. This modules allow to access on-line to parallel computer systems for the execution of the parallel mapping algorithms (under construction).

2. Algorithms of mapping a graph of a parallel program into a graph of a parallel computing system and data structures

We consider parallel programs consisting of modules that can work and exchange information simultaneously. Such a program is represented by a


File Edit View Go Communicator Help
Reload Home Search Guide Print Security
Bookmarks Go to: http://rav.sscc.ru/~monakhov/topas.html
System: Edit system
Program: Edit program
Neuro algorithm
Distributed neuro algorithm
Advanced neuro algorithm
Simulated annealing
Local simulated annealing
Advanced simulated annealing
Evolutionary method
Genetic method
Location method

The introduction

TOPAS (Test and Optimization of Parallel Algorithms and Structures), a programming tool for visualization, animation and investigation of algorithms of mapping graphs is presented. The tool is implemented in Java and can be accessed on WWW.

About TOPAS...

Methods for automatic mapping of a graph structure into another graph structure can be used for the mapping of modules of parallel programs into processors of parallel computing systems, as well as for special data decompositions and solving the problems of operations research, graph theory, combinatorial optimizations, networking, etc. In other word, mapping algorithms are important for various applications. The system supports visualization and animation of graphs, mapping processes, as well as visualization of mapping results. The system includes libraries of sequential and parallel mapping algorithms based on neuron networks, evolutionary and genetic modeling, simulated annealing, local and hierarchical optimization. In addition, it can be used to define some "atomic" operations (for example, partitioning of the complex data structures, e.g. irregular meshes) for more complex problems. The system has been written in Java and is an applet, that allows to place it into an HTML-page and to demonstrate algorithms at any point of the world where Internet is accessible. We call this system as TOPAS (Test and Optimization of Parallel Algorithms

Figure 1. The cover frame of the TOPAS system

weighted graph, where nodes mean modules and edges are information links. Also we represent parallel computing systems by weighted graphs in which nodes are processors and edges are interconnections.

A graph mapping algorithm is to find out a mapping of one graph into another which gives the minimal time of working of a parallel program on a parallel system. Now in TOPAS the nine algorithms are presented (Figure 1):

- two of them are the evolution [3] and the genetic algorithms [6],
- three of them are the methods of simulated annealing [6],
- three of them use the Kohonen neural networks [4],
- one is the local optimization algorithm [5],
- and one is the hierarchical decomposition and mapping algorithms [7] (under construction).

The goal of the mapping algorithms is to produce an allocation of the modules of a parallel program into processors of a system with minimum of the objective cost function. The objective function represents the interprocessor communication cost and computational load balance of the processors during execution of the program. A specific type of the objective function can be chosen in the window of the mapping algorithm parameters.

3. Visual components of the system

3.1. Graph editor. The graph editor is actually an independent part of the system. It can be used in any applet where dynamic graph editing is

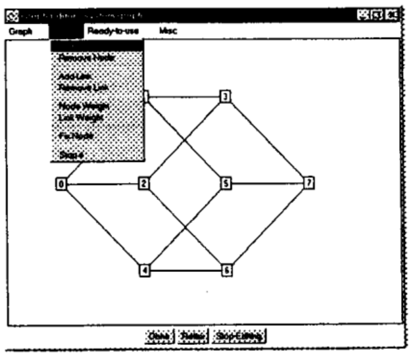


Figure 2. The **GraphEditor** representation frame: system graph

needed. It is implemented in **GraphEditor** class which expands **Frame** class. To start the editor it is necessary to create an instance of **GraphEditor** class by giving an object of **Graph** class which requires editing to its constructor.

The **Graph** class represents structures of weighted graphs. It also contains some additional information about subsystem's modules to be used. An object of **Graph**, thus, describes a particular weighed graph. **Node** and **Edge** are nodes and edges classes of this graph, respectively.

After creating an instance of **GraphEditor**, a window of the graph editor appears on the screen (Figure 2). It consists of three parts: a basic menu, a working area and a tool bar. A graph is displayed in the working area and all manipulations on it (moving nodes, removing and creating them, removing and creating edges, changing their weights, etc.) are made there. To display regular graphs with a big number of nodes the parametric representation of the graphs is used.

The tool bar is located under the working area. There can be three buttons: *Done*, *Relax*, and *Stop Editing*. Pressing on the *Done* button means that the work is completed (it stops modifying the graph and the working of **GraphEditor**). The *Relax* button switches on and off a "spring graph" mode. In this mode the graph behaves so as if it consists of weightless nodes connected to springs in very "viscous" environment. The *Stop Editing* button turns off an editing graph mode which is turned on through the basic menu.

The basic menu supports various manipulations on the graph:

1. **Graph** menu item:
   - *New* button clears a current graph and makes it empty;
   - *Load* button loads a graph from a file;
   - *Merge* button adds a graph from a file to a current graph;
   - *Save* button saves a current graph;

- *Save as ...* button saves a current graph into a given file;
- *Done* button finishes the work of the graph editor.

Note: the operations with files are possible only on a local machine or if a web-browser allows to work with a disk.

2. **Edit** menu item (see Figure 2):

   - *Add Node* button turns on the mode of nodes adding: by pressing the mouse button with the cursor in the working area, a new graph node will appear;
   - *Remove Node* button turns on the mode of nodes removing;
   - *Add Link* button supports a mode of edges adding;
   - *Remove Link* button is for edges removing;
   - *Node Weight* button is for change of nodes' weights;
   - *Link Weight* button is for change of edges' weights;
   - *Fix Node* button allows to fix and to release nodes. A fixed node cannot move in the "spring graph" mode;
   - *Stop* button turns off any edit mode.

3. *Ready-to-use* menu item allows to insert a subgraph of one of six pre-defined types. After choosing an item from this menu there appears a dialog window offering to set parameters necessary for this subgraph. The supported types now are the following: a set of disconnected nodes, lines, rings, grids, hypercubes and fully-connected structures.
4. **Misc** menu item:

   - *Default weights* button sets the default weights of nodes and edges, for example, at the addition of nodes or edges in the graph;
   - *Recalc lengths* button recalculates lengths of springs so that in the "spring graph" mode the current state of the graph becomes steady.

3.2. Graph icons. Graph icons are just objects each of which keeps a reference to a graph object and displays all its changes.

3.3. Supervising of the execution and animation algorithm. The supervising module can also be used independently. It is implemented in `RunningAlgorithm` class which extends `Frame` class. The user should create an instance of this class by giving to its constructor two objects of `Graph` class (initial data, system and program graphs) and an object of `Method` class (which describes an algorithm to be used). `Graph` and `Method` classes are described in Section 5. After the initialization a window appears on

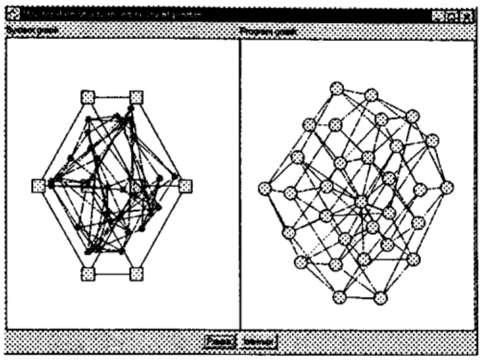


Figure 3. The frame for animation of mapping algorithm

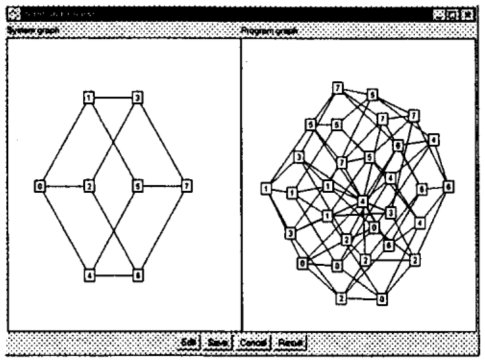


Figure 4. The frame for representation of results

the screen. This window consists of three fields: a system graph panel, a program graph panel and a control panel. On the program graph panel, a program graph is displayed. On the system graph panel, the animation of the mapping algorithm execution is displayed (Figure 3). The control panel has two buttons: *Run* and *Interrupt.* The first one runs an algorithm and changes the button caption. The second one pauses the algorithm. For the algorithm being paused, the user can check intermediate results: clicking on a processor node in the system graph panel marks all processes that are now put into this processor, and clicking on a process node in the program graph panel shows a processor on which this process is mapped as well as all processes mapped on it. The Interrupt button stops the algorithm execution.

After the finishing of the algorithm execution, the representing and saving results module begins a job. The module is also based on a window with two panels for graphs' images and a control panel (Figure 4). Nodes of the system graph are numbered by their orders and nodes of the program graphs are marked by numbers of processors on which they are mapped.

In addition, processors are painted with different colors, and processes are correspondingly painted, too.

The control panel has four buttons: *Edit*, *Save*, *Cancel*, and *Result*. *Edit* button allows to correct the mapping by hands in the graphical mode, *Save* allows to save results, *Cancel* closes the window without saving, and *Result* shows the information (which will be saved) in a new window.

4. Non-visual components of the system

4.1. Graph structure. The structures which represent graphs of systems and programs are defined in the `Graph` class. It also contains functions for nodes and edges manipulations (adding and removing), for calculation of a matrix of shortest distances between nodes. The `Node` and `Edge` classes are used here and describe the corresponding elements of the graph.

4.2. Programmer interface. To add a new method into the system, a programmer should create a new class which extends the `Method` interface. This interface contains functions that should be defined by the programmer. The `Iteration` function describes one step of the algorithm and `InitMethod` function initializes data structures needed for the algorithm to work and gets two graphs that are an initial data.

5. Conclusion

TOPAS is just a part of the project oriented on Web. The system provides a convenient visual interface and special embedded knowledge about mapping algorithms. TOPAS supports research and development, visualization and animation, testing and debugging of new algorithms for the mapping of the parallel program graphs onto multicomputer systems. In addition, it is also very suitable for using of mapping algorithms as elementary operations for specifications of a sophisticated data decomposition and for more complex applications.

Acknowledgements. The author would like to thank Prof. N.N. Mirenkov for fruitful discussions, O. Chunikhin, E. Grosbein, A. Musin, T. Gurnenko, and A. Serbin for assistance in implementation of this project.

References

[1] N. Mirenkov, *VIM language paradigm*, Lecture Notes in Computer Science, ed. by B. Buchberger, J. Volkert, Springer-Verlag, **854**, 1994, 569–580.

[2] S. Bridgeman, A. Gard, and R. Tamassia, *A graph drawing and translation service on the WWW*, Graph Drawing'96 (Proc. GD'96). LNCS, **1190**, 1996, 45–52.

[3] O.G. Monakhov and E.B. Grosbein, *A parallel evolution algorithm for graph mapping problem*, Proc. Inter. Workshop on Parallel Computation and Scheduling (PCS'97), CICESE, Ensenada, Baja California, Mexico, 1997, 17–21.

[4] O.G. Monakhov and O.J. Chunikhin, *Parallel mapping of program graphs into parallel computers by self-organization algorithm*, Applied Parallel Computing (Proc. PARA'96). LNCS, **1184**, 1996, 525–528.

[5] O.G. Monakhov, *Parallel mapping of parallel program graphs into parallel computers*, Proc. Int. Conf. Parallel Computing-91, Elsevier Science Publishers, Amsterdam, 1992, 413–418.

[6] O.G. Monakhov, E.B. Grosbein, and A.R. Musin, *Mapping algorithms for parallel systems based on evolutionary modelling and simulated annealing*, Proc. Inter. Workshop on Distributed Data Processing (DDP'98), Novosibirsk, Russia, 1998, 142–146 (in Russian).

[7] O.G. Monakhov, *Parallel algorithm for mapping parallel programs into pyramidal multiprocessor*, Applied Parallel Computing (Proc. PARA'95). LNCS, **1041**, Springer, 1995, 436–442.