TOPAS: a Web-based Tool for Visualization of Mapping Algorithms

0. G. Monakhov, 0. J. Chunikhin, E. B. Grosbein

Institute of Computational Mathematics and Mathematical Geophysics,
Siberian Division of Russian Academy of Sciences

Abstract

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 is accessed on WWW: htip://rav.sscc.ru/~monakhov/topas.html.

Keywords: *parallel algorithms, mapping, multicomputer system, visualisation of methods, Java, network technologies, programming environment.*

1 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 word, 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 [I]. The system has been written in Java and is an applet, that allows to place it into an HTML-page (http://rav.sscc.ru/~monakhov/topas.html) and to demonstrate algorithms at any point of the world where Internet is accessible (this approach is analogous to [2]). We call this system as TOPAS (Test and Optimization of Parallel Algorithms and Structures). In general, it is related to the development of a problem-solving environment which integrates multimedia, parallel and distributed Internet technologies for specification (programming) of application algorithms, and the control of mapping strategies and data visualization.

Different parts of the subsystem (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.

2 Structure of TOPAS

The subsystem consists of the following parts:

1. *Core.* The TOPAS's core is an applet, which displays initial data, calls the graph editor (for data editing) and starts necessary algorithms.
2. *Graph's icons.* They are used by the core to display the initial data and are small images of the graphs.
3. *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.
4. *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.
5. *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.
6. *Libraries of sequental and parallel mapping algorithms.*
7. *Help module.* This module displays information about system, interfaces, problem and mapping algorithms.
8. *Internal representation of graph structure.* This format allows to connect all components of the system.
9. *Programmer's interface.* It allows to easily insert new algorithms into the system.
10. *Distributed agent system.* This modules allow to access on-line to parallel computer systems for the execution of the parallel mapping algorithms (under consiraction).

3 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 which can work simultaneously and exchange information. Such a program is represented by a 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 mapping one graph inio another which gives minimal time of working a parallel program on a parallel system. Now in TOPAS nine algorithms are presented (Fig.I):

- two are evolution [3] and genetic algorithms,
- three are methods of simulated annealing,
- three use Kohonen neural networks [4]
- a local optimization algorithm [5].

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 interprocess or 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.

4 Visual components of the system 4.1

Graph editor

The graph editor is actually an independent part of the subsystem. It can be used in any applet where dynamic graph editing is used. It is implemented in a class GraphEditor which expands class Frame. To start the editor it is necessary to create a copy of a class GraphEditor by giving an object of a class Graph which requires editing to its constructor.

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

After creating a GraphEditor copy a window of the graph editor appears on the screen (Fig.2). It consists of three parts: a basic menu, a working area and a toolbar. A graph is displayed in the working area and all manipulations of 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 toolbar is located under the working area. There can be three buttons: Done, Relax and Stop Editing. Pressing on button Done means that the work is completed (it stops modifying the graph and the working GraphEditor). Button Relax switches on and off a mode "spring graph". In this mode the graph behaves so as if it consists of weightless nodes connected to springs in very "viscous" environment. Button Stop Editing switches off a mode of editing the graph which is switched on through the basic menu.

The basic menu supports various manipulations of the graph:

1. Menu item **Graph**

   New clears a current graph and makes itempty.

   Load loads a graph from a file.

   Merge adds a graph from a file to a current graph.

   Save saves a current graph.

   Save as ... saves a current graph in a file of a given name.

   Done 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. Menu item **Edit** (Fig.2)

   Add Node switches on a mode of adding nodes: at pressing the mouse button with the cursor in the working area, there will appear a new node of the graph.

   Remove Node switches on a mode of removing nodes.

   Add Link supports a mode of adding edges.

   Remove Link is for removing edges.

   Node Weight is for changing nodes' weights.

   Link Weight is for changing edges' weights.

   Fix Node allows to fix and to release nodes. A fixed node cannot move in mode "spring graph".

   Stop switches off any edit mode.

3. Menu item **Ready-to-use** allows to insert a subgraph of one of six predefined types. After choosing an item from this menu there appears a dialog window offering to set parameters necessary for this subgraph. Now six types are the following: a set of disconnected nodes, lines, rings, grids, hypcrcubes and fully-connected structures.

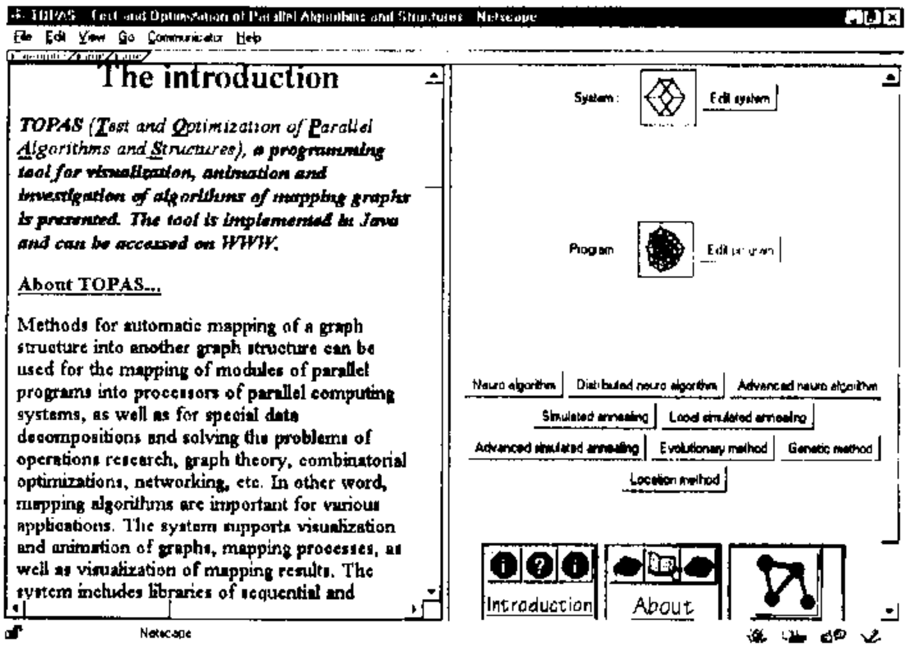
4. Menu item **Misc**

   Default weights sets weights of nodes and edges used by default, for example**,** at the addition of nodes or edges in the graph.

   Recalc lengths recalculates lengths of springs so that in mode "spring graph" the current state of the graph becomes steady.

4.2 Graphs' icons

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


TOPAS - Test and Optimization of Parallel Algorithms and Structures - Netscape
File Edit View Go Communicator Help
System :
Edit system
Program
Edit program
Neuro algorithms
Distributed neuro algorithms
Advanced neuro algorithm
Simulated annealing
Local simulated annealing
Advanced simulated annealing
Evolutionary method
Genetic method
Location method
Introduction
About
Netscape

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

Figure I: The cover frame of the TOPAS system

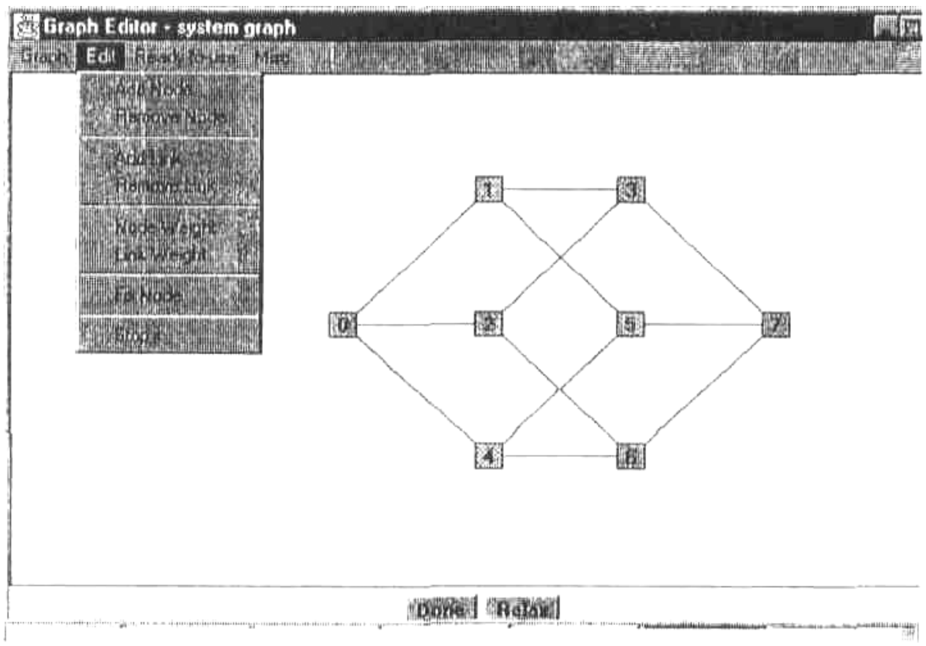


Figure 2: The frame for representation of GraphEditor: system graph

Moscow, September 7-11

4.3 Supervising the algorithm execution and animation

The supervising module also can be used independently. It is implemented in class RunningAlgorithm which extends class Frame. The user is to create an object of this class by giving to its constructor two objects of class Graph (initial data, system and program graphs) and an object of class Method (which describes an algorithm to be used). Classes Graph and Method will be described in section 5. After an initializing operation a window appears on 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 (Fig.3). The control panel has two buttons: Run and Interrupt. The first one is to run an algorithm and lo change the button caption. The second one is to pause 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 finishing the algorithm execution, the module of representing and saving results begins working. The module is also based on a window with two panels for graphs' images and a control panel (Fig.4). Nodes of the system graph are numbered by their orders and nodes of the program graphs are marked with 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 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.

5 Non-visual components of the system

5.1 Graph structure

Structures representing the graphs of systems and programs are defined in the class Graph. This class also contains functions for adding and removing nodes and edges, calculating the matrix of shortest distances between nodes. There are also used classes Node and Edge that describe the corresponding elements of the graph.

5.2 Programmer's interface

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

6 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 new algorithms of mapping of the parallel program graphs onto multicomputer systems. In addition, it is also very suitable for making use of mapping algorithms as elementary operations for specifications of sophisticated data decompositions and more complex application methods.

References

[1] N. Mirenkov. VIM Language Paradigm, in: *Lecture Notes in Computer Science,* 854. B. Buchberger, J. Volkert (Eds.), Springer-Verlag, 1994, 569-580.

[2] S. Bridgeman, A. Card, R. Tamassia, A graph drawing and translation service on the WWW, in: *Graph Draw-ing'96 (Proc. GD'96). LNCS.voi.ll90,* 1996.45-52.

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

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

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

Author(s):

Dr. Oleg G. Monakhov, the principal scientist of Institute of Computational Mathematics and Mathematical Geophysics, Siberian Division of Russian Academy of Sciences.

Oleg Chunikhin, student of Novosibirsk State University.

Eugene Grosbein, student of Novosibirsk State University.

Address: Pr. Lavrentieva, 6, Novosibirsk, 630090, Russia

Phone: +7-3832-341066, Fax: +7-3832-324259

E-mail: monakhov@rav.sscc.ru

Figure 3: The frame for animation of mapping algorithm

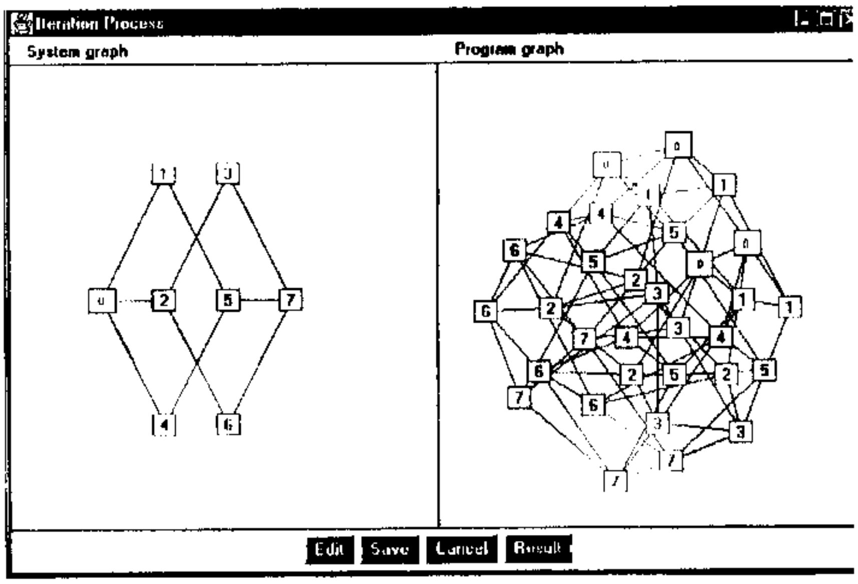


Figure 4: The frame for representation of results