Methods and Tools for Support of Graphs and Visual Processing

VICTOR KASYANOV
Laboratory for Program Construction and Optimization
A.P. Ershov Institute of Informatics Systems
630090, Novosibirsk, Lavrentiev pr. 6
RUSSIA

Abstract: The main thesis of this paper is that intricate nature of software systems can, and in our opinion should, be represented by graph models. Graphs are used in computer science almost everywhere, since graph is a very natural way of explaining complex situations on an intuitive level.
In the paper, methods and systems for visual processing of graphs and graph models are presented. Our book series intended to be a general guide on graph algorithms is outlined. A dictionary of graph theory in computer science and its electronic version are described.

Key-Words: Hierarchical graphs, graph algorithms, graph models, visual processing, dictionary of graph theory.

1 Introduction

Graph is the most common "abstract" structure encountered in computer science and computer education. Any system that consists of discrete states (or sites) and connections between them can be modeled by a graph. Graphs are used in computer science almost everywhere, e.g. as data and control flow diagrams, entity relationship diagrams, Petri nets, visualization of software and hardware architectures, evaluation diagrams of nondeterministic processes, SADT diagrams and many more.

The main thesis underlying this paper is that intricate nature of software systems can, and in our opinion should, be represented by graph models. Their wide applicability is due to the fact that graphs are a very natural way of explaining complex situations on an intuitive level.

We are entirely convinced the future is 'visual', and the graph models are the best formalism for visual representation of information of complex and intricate nature. Visualization of a conceptual structure is a key component of support tools for complex applications in science and engineering. Moreover, the information that is interesting us in computer science is of structural and relational rather than quantitative nature.

Many modern software systems, in particular for graphics workstations, include graph drawing and visual processing functions. A bibliography on graph drawing [1] cites more than 300 papers written before 1993, and from 1993 the problem is the subject of the annual conference with over 100 participants.

In the paper, we present our projects on the development of methods and tools to support graphs and visual processing. We begin with consideration of methods and tools for visual processing of graphs and graph models. Then we outline our series of books intended to be a general guide on graph algorithms. Finally, we summarize our results on construction of a dictionary of graph theory in computer science and its electronic version.

The work is partially supported by the Russian Foundation for Basic Research under grant RFBR 00-07-90296 and by the Ministry on Higher Education of Russia under state program “Universities of Russia”

2 Methods and systems for visual processing

Graph models can be used in practice only along with support tools that provide visualization, editing and processing of graphs and graph models. For this reason many graph visualization systems, graph editors and libraries of graph algorithms have been developed in recent years. Examples of these tools include VCG [14], daVinci [3], Graphlet [5], GraVis [11], GLT&GET [12] and LEDA [13].

In some application areas, organization of information is too complex to be modeled by a classical graph. To represent a hierarchical kind of diagramming objects, more powerful graph formalisms have been introduced, e.g. higraphs [4], compound digraphs [15] and clustered graphs [2]. One of the recent nonsclassical graph formalisms is hierarchical graphs and graph models [7].

2.1 Hierarchical graphs and graph models

A *hierarchical graph H=(G,T)* consists of an *underlying* graph *G* and an *inclusion* tree *T*. The underlying graph *G* can be any undirected graph, digraph or hypergraph. The inclusion tree *T* represents a recursive partitioning of G into fragments.

In Fig. 1 we can see an example of a hierarchical graph *H=(G,T)* with only two nontrivial fragments *C* and *G* as well as an example of a hierarchical graph drawing *D* on the plane where every nontrivial fragment of *G* is represented as a rectangle.

H is a *simple* hierarchical graph if all fragments of *H* are induced subgraphs of *G*. *H* is a *connected* hierarchical graph if each fragment of *H* is connected graph. It should be noted that any clustered graph *H* can be considered as a simple hierarchical graph *H=(G, T),* such that *G* is an undirected graph and the leaves of *T* are exactly the trivial subgraphs of *G*.

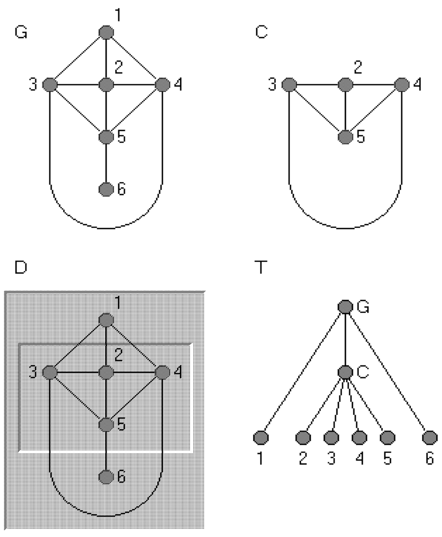


Fig. 1

A *drawing D* of a hierarchical graph *H* is a representation of *H* in the plane such that the following properties hold (See Fig. 1). Each vertex of *G* is represented either by a point or by a simple closed region. The region is defined by its *boundary* - a simple closed curve in the plane. Each fragment of *G* is drawn as a simple closed region which include all vertices and subfragments of the fragment. Each edge of *G* is represented by a simple curve between the drawing of its endpoints.

D is a *structural* drawing of *H* if all edges of any fragment of *H* are located inside the region of the fragment. In Fig.1 we can see an example of nonsructural drawing of hierarchical graph.

A hierarchical graph is a *planar* one if it has such a structural drawing that there are no crossing between distinct edges and the boundaries of distinct fragments.

The following properties hold [7, 8].

Theorem 1. There are nonplanar hierarchical graphs with planar underlying graphs.

Theorem 2. There are nonplanar hierarchical graphs having nonstructural planar drawing.

Theorem 3. A simple connected hierarchical graph *H=(G,T)* is a planar graph if and only if there is such a planar drawing *D* of *G* that for any vertex *p* of *T* all vertices and edges of *G-G(p)* are in the outer face of the drawing of *G(p)*.

A *hierarchical graph model* is defined as a set of labelled hierarchical graphs together with an equivalence relation between them. The equivalence relation can be specified in different ways, e.g. it can be defined via invariants (i.e. properties being inherent in equivalent labelled graphs) or by means of so-called equivalent transformations that preserve the invariants.

2.2 The HIGRES and VEGRAS systems

Hierarchical graphs and graph models can be used in many areas where strong structuring and visualization of information is needed [7, 8, 10]. Several general-purpose graph visualization systems provide recursive folding of subgraphs allowing to create clusters. However, this feature is commonly used only to hide a part of information and not always applicable to visualization of hierarchical graphs. Usual graph editors either do not have any support to attributed graphs or have a rather weak support. Though the GML file format used by Graphlet [5] can store arbitrary number of labels associated with each graph element, it is impossible to edit and visualize these labels in Graphlet's graph editor. Most editors allow only one text label for each vertex and optionally for each edge.

We present the HIGRES and VEGRAS systems that support visualizing, editing and processing of hierarchical graphs and graph models. The systems are implemented in C++ and work under Microsoft Windows 95/98/NT.

The HIGRES system is a visualization tool and an editor for attributed hierarchical graphs and a platform for execution and animation of graph algorithms.

A hierarchical graph supported by the HIGRES system consists of labelled vertices, fragments and edges, which are considered as objects. Vertices and edges form an underlying graph. Every fragment is represented by a rectangle and can be closed or open. When a fragment is open, its content is visible; when it is closed, it is drawn as an empty rectangle with labels inside. A separate window can be opened for each fragment.

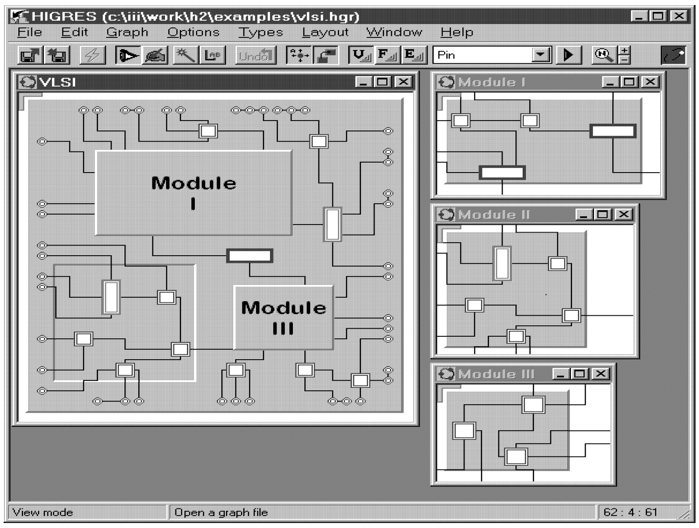


Fig. 2

In Fig. 2 we can see a VLSI model represented by a hierarchical graph. The model consists of three modules folded into fragments. These fragments are displayed in separate windows with different scales. One fragment (Module II) is open. Its content is also displayed inside the VLSI window. Two other fragments are closed and shown in this window as covered rectangles. Type of an object defines the most part of its visual attributes. This means that semantically relative objects have similar visual representation.

Type of an object defines the most part of its visual attributes. This means that semantically relative objects have similar visual representation. The HIGRES system uses a flexible technique to visualize object labels. The user specifies a text template for each object type. This template is used to create the label text of objects of the given type by inserting labels' values of an object. The user can create new object types and labels.

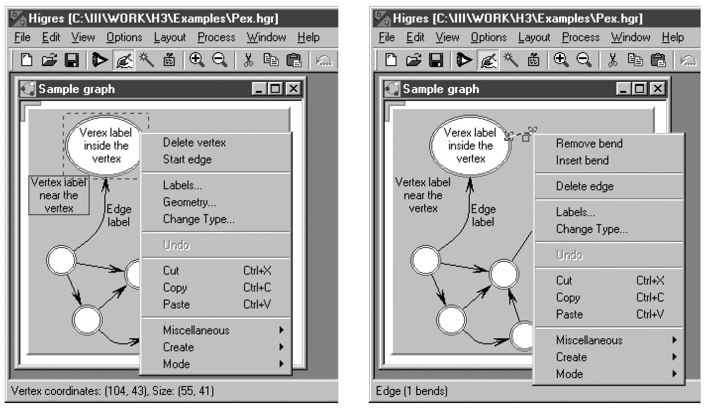

Higres [C:\III\WORK\H3\Examples\Pex.hgr]
File Edit View Options Layout Process Window Help
Sample graph
Verex label inside the vertex
Vertex label near the vertex
Edge label
Delete vertex
Start edge
Labels...
Geometry...
Change Type...
Undo
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Miscellaneous
Create
Mode
Vertex coordinates: (104, 43), Size: (55, 41)
Remove bend
Insert bend
Delete edge
Edge (1 bends)

Fig. 3.

Other visualization features are the following:

- various shapes and styles for vertices,
- both polyline and smooth curved edges,
- various styles for edge lines and arrows,
- color selection for all graph components, possibility to scale graph image to arbitrary size,
- edge text movable along the edge line, external vertex text movable around the vertex,
- font selection for label text,
- two graphical output formats,
- a number of options that control graph visualization.

The system uses two basic modes: view and edit. In the view mode it is possible only to open/close fragments and fragment windows, but the scrolling operations are extended with mouse scrolling.

In the edit mode the left mouse button is used to select objects and the right mouse button displays the pop-up menu, in which the user can choose the operation he/she wants to perform (See Fig. 3).

It is also possible to create new objects by selecting commands in this menu. The left mouse button can be also used to move vertices, fragments, label texts and edge bends, and resize vertices and fragments. All edit operations are gathered in a single edit mode. In our opinion, it is a more useful approach (especially for inexperienced users) than division into several modes. However, for adherents of the latter case we provide two additional modes. Their usage is optional but in some cases they may be useful: the "creation" mode for object creation and "labels" mode for label editing.

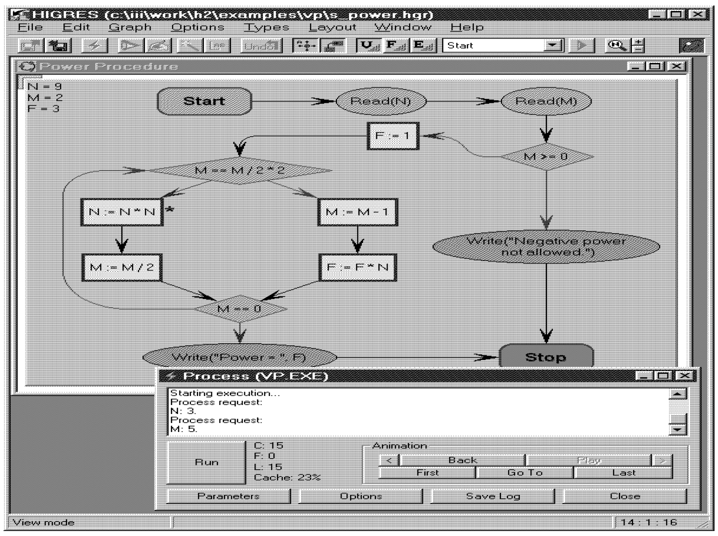
HIGRES has a user interface with

- almost unlimited number of undo levels,
- optimized screen update,
- automatic elimination of object overlapping,
- automatic vertex size adjusting,
- grid with several parameters,
- a number of options that configure user interface,
- online help available for each menu,
- dialog box and editor mode.

The system HIGRES supports construction and execution of a wide range of algorithms for visual processing of hierarchical graph models. In particular, it provides a run-time, repeated and backward animation of graph algorithms.

As examples in the paper we consider tree algorithms implemented in the framework of the HIGRES system. They can be used to visualize the work of a given finite automaton, Petri net and program scheme.

The work of a graph algorithm is illustrated in Fig.4 where a program scheme interpreted by an external module is presented. Here asterisk shows which operator has control. Current values of variables are listed in the top left corner of the fragment rectangle. This screenshot was made after the completion of the process and rewinding several samples back.



```
Starting execution...
Process request:
N: 3.
Process request:
M: 5.
```

```
C: 15
F: 0
L: 15
Cache: 23%
```

Fig. 4

The HIGRES system provides a special C++ API that can be used to create external modules. This API includes functions for graph modification and functions that provide interaction with the system. It is unnecessary for a programmer who uses this API to know the details of the internal representation of graphs. Hence, the creation of new modules is a rather simple work.

The system HIGRES is available at Web [16].

VEGRAS is a universal and simple-to-use editor of attributed graphs, including hierarchical, oriented to support of construction of qualitative graph illustrations (See Fig. 5). VEGRAS also supports exchange of the graph illustrations with other Windows applications, including the HIGRES system.

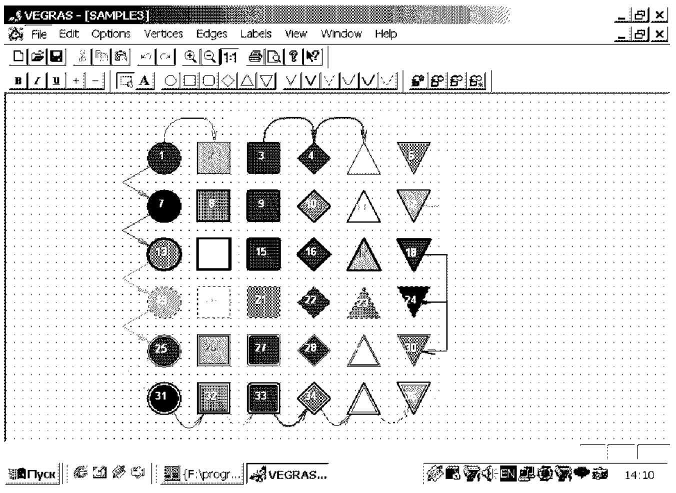


Fig. 5

3 A guide on graph algorithms

As a rule a specific problem can be solved on the base of known efficient graph processing algorithms and methods, but to find (and understand) the needed algorithms among a great number of scientific papers can be an unsolvable problem for an end-user. We believe that a general guide on graph algorithms would be useful for programmers and decided to prepare a series of books intended to be not only a reference manual of graph algorithms but also an introduction to the part of the graph theory and its applications to computer science and programming. We thought that it is reasonable to group graph algorithms into certain classes which deal with the same type of graphs.

In contrast to Donald Knuth who used the assembly language of the so-called MIX computer in his fundamental books "The art of computer programming", we decided to use in our book series a high-level and language-independent representation of graph algorithms. In our view, such an approach is preferential, as it allows us to describe algorithms in a form that admits direct analysis of their correctness and complexity, as well as a simple translation of algorithms to high-level programming languages without disturbing correctness and complexity. Besides, this approach to the algorithm presentation allows the readers to understand an algorithm at the informative level, to evaluate its applicability to a specific problem, and to make all its modifications needed for correct application of the algorithm.

At present, the series consists of three books: "Graph theory: algorithms for processing trees" (1994), "Graph theory: algorithms for processing acyclic graphs" (1998) and "Reducible graphs and graph models in programming" (1999).

The first book contains a high-level and language-independent description of the methods and algorithms on trees, the most important type of graphs in computer science. The book includes algorithms for traversals and generation of trees, finding spanning trees, computing of structural trees, isomorphism of trees, unification and transformations of trees, using trees for search and retrieval of information, trees as data structures, using trees in syntax analysis.

The book consists of three parts. In the first part, we present the main notions, properties of trees, and some basic algorithms on trees, such as depth-first and breadth-first traversals of trees, the algorithms of coding and generation of trees, etc. The second part deals with applications of trees to problems connected with structuring of programs, unification problem, term rewriting systems, syntax analysis, etc. The third part is devoted to the problems of data storage and retrieval.

This book has been translated into English [9].

The second book is devoted to the directed acyclic graphs (or DAGs) which simulate posets and, like trees, form an important class of graphs that is widely used in computer science and programming. In this book, some basic techniques and algorithms connected to different applications of DAGs to computer science are considered. Then, the elements of the theory of posets, lattices and semilattices are given. Finally, algorithms for the semantic analysis and the object code generation are presented.

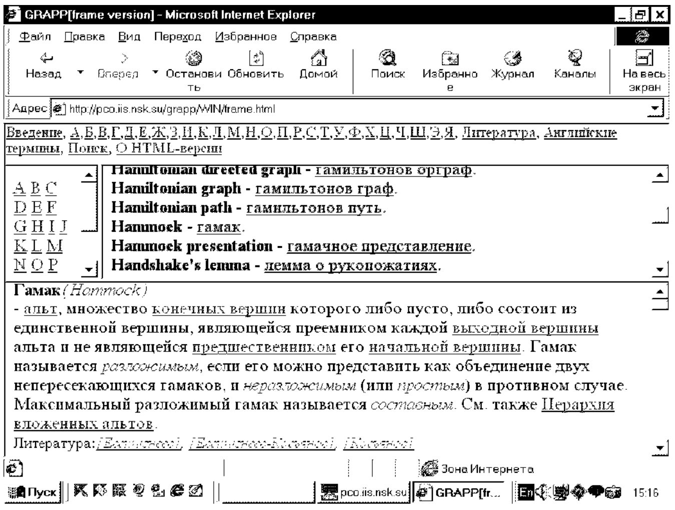
The book "Reducible graphs and graph models in programming" consists of two parts.

The first part is dedicated to the class of algorithms for reducible (or regularizaible) graphs that expand DAGs and are the most common graph models of the structured programs. This class of graphs plays a very important role in software systems, e.g. many compiler optimizations are simpler, more efficient or applicable when the control flow graph is reducible.

In the second part of the book, some graph models widely used in computer science (such as program schemata, Petri nets, intermediate program representations, etc) are considered. Here, the main attention is given to such subjects as program optimization, automatic parallelization, visual processing, simulation of the parallel and distributed systems, structural complexity of programs, etc.

4 Dictionary on graph theory in computer science and programming

The problem of terminology is one of the main problems in application of graph methods to programming and computer science. The dictionary [6] recently published is the first attempt to solve this problem. The most common terms on graph theory and its applications to computer science and programming are collected in it. The articles of the dictionary are supplied with illustrations, cross-references and references to the available literature. The English equivalents of the terms allow the reader to use the dictionary while translating from Russian into English and back.


GRAPP[frame version] - Microsoft Internet Explorer
Файл Правка Вид Переход Избранное Справка
Назад
Вперед
Остановить
Обновить
Домой
Поиск
Избранное
Журнал
Каналы
На весь экран
Адрес
http://pco.iis.nsk.su/grapp/WIN/frame.html
Зона Интернета
Пуск
pco.iis.nsk.su
GRAPP[fr...
En
15:16

Введение, А,Б,В,Г,Д,Е,Ж,З,И,К,Л,М,Н,О,П,Р,С,Т,У,Ф,Х,Ц,Ч,Ш,Э,Я, Литература, Английские термины, Поиск, О HTML-версии

A B C
D E F
G H I J
K L M
N O P

Hamiltonian directed graph - гамильтонов орграф.
Hamiltonian graph - гамильтонов граф.
Hamiltonian path - гамильтонов путь.
Hammock - гамак.
Hammock presentation - гамачное представление.
Handshake's lemma - лемма о рукопожатиях.

Гамак *(Hammock)*
- альт, множество конечных вершин которого либо пусто, либо состоит из единственной вершины, являющейся преемником каждой выходной вершины альта и не являющейся предшественником его начальной вершины. Гамак называется *разложимым*, если его можно представить как объединение двух непересекающихся гамаков, и *неразложимым* (или *простым*) в противном случае. Максимальный разложимый гамак называется *составным*. См. также Иерархия вложенных альтов.
Литература: [illegible]

Fig. 6

The preliminary version of the dictionary was published in 1995-96 in the Novosibirsk State University. The Web-dictionary based on this version is named GRAPP (See. Fig. 6) and is available at Web [16].

5 Conclusion

The use of visualization methodology based on graphs seems to be a very interesting approach to teaching and learning computer science. In the paper some recent results of the author and his collaborators on methods and tools for graph support in computer science have been considered.

The author is grateful to all colleagues from the A. P. Ershov Institute of Informatics Systems and the Novosibirsk State University who took part in the projects considered in the paper, first of all, to Prof. V.A. Evsigneev, and also to I.A. Lisitsyn, T.S. Merdischeva, E.S. Merdischeva and V.E. Kazantzev.

References:

[1] G. Di Battista, P. Eades, R. Tamassia, I.G. Tollis. Algorithms for drawing graphs: an annotated bibliography, *Computational Geometry: Theory and Applications*, Vol. 4, No 5, 1994, pp. 235-282.

[2] Q. W. Feng, R.F. Cohen, P. Eades, Planarity for clustered graphs, *Lecture Notes in Computer Science.,* Springer Verlag, Vol.979, 1995, pp. 213 - 226.

[3] M. Frцhlich, M. Werner. Demonstration of the interactive graph visualization system daVinci, *Lecture Notes in Computer Science*, Springer Verlag, Vol. 894, 1994, pp.266-269.

[4] D. Harel, On visual formalism, *Commun. ACM,* Vol.31, No 5, 1988, pp.514 - 530.

[5] M. Himsolt. The Graphlet system (system demonstration), *Lecture Notes in Computer Science,* Springer Verlag, Vol.1190, 1996, pp. 233-240.

[6] V.A. Evstigneev, V.N. Kasyanov. *Explanatory Dictionary on Graph Theory in Computer Science and Programming,* Nauka Publ., Novosibirsk, 1999, 288 p. (In Russian).

[7] V.N. Kasyanov. Hierarchical graphs and visual processing, In: *Intern. Congress of Mathematicians (ICM98). Abstracts of Short Communications and Poster Sessions*, Berlin, 1998, P. 292.

[8] V. N. Kasyanov. Graph applications in programming, *Programmirovanie*, No.3, 2001, pp. 51-70. (In Russian)

[9] V.N. Kasyanov, V.A. Evstigneev. *Graph theory for programmers. Algorithms for processing trees*, Kluwer Academic Publishers, 2000, 432 p.

[10] V.N. Kasyanov, I.A. Lisitsyn. Hierarchical graph models and visual processing, In *Proceedings of Conference on Software: Theory and Practice. 16th IFIP World Computer Congress 2000*, Beijing, 2000, pp. 179-182.

[11] H. Lauer, M. Ettrich, K. Soukup. GraVis - System Demonstration, *Lecture Notes in Computer Science*, Springer Verlag, Vol.1353, 1997, pp.344-349.

[12] B. Madden, P. Madden, S. Powers, M. Himsolt. Portable Graph Layout and Editing, *Lecture Notes in Computer Science,* Springer Verlag, Vol. 1027, 1995, pp.385-395.

[13] K. Mehlhorn, S. Nrher, LEDA: a platform for combinatorial and geometric computing, *Comm. ACM*, Vol. 38, 1995, pp.96-102.

[14] G. Sander, Graph layout through the VCG tool, *Lecture Notes in Computer Science*, Springer Verlag, Vol. 894, 1994, pp. 194-205.

[15] K. Sugiyama, K. Misue, Visualization of structured digraphs, *IEEE Trans. on Systems, Man and Cybernatics*, Vol. 21, No 4, 1991, pp. 876-892.

[16] The GRAPP system is available at Web from <http: // pco.iis.nsk.su/grapp>.

[17] The HIGRES system is available at Web from <http: // pco.iis.nsk.su/higres>.