БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ . BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, 55
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, 55

София . 2005 . Sofia

Graph and Network Optimization Software Package*

Stanislav Drangajov, Vassil Vassilev, Nikolay Dimitrov, Mariana Djelatova

Institute of Information Technologies, 1113 Sofia

1. Introduction

With the development of the PC, network technology and respective operating systems, a variety of products – initially under DOS and later – under WINDOWS or Linux, and in the global network were developed for network and graph optimization. Some well-known systems concerning these problems, are GIDEN [3], GRIN [4], GOBLIN [5], LEDA [6], WinQBS [7]. This paper does not consider some specially designed systems on main frames, dedicated to solve very large scale problems.

GIDEN is oriented towards network optimization. It has a very well developed graphical user interface. It proposes several fixed types of algorithms (shortest path, max-flow, min cost flow, network simplex etc.). GRIN is directed mainly to solving problems on graphs (coloring, spanning trees etc.) and just a limited set of optimization problems on networks (e.g. shortest path, max flow). The graphical interface is good. Tabular representation is realized by adjacency matrix where only one parameter may be specified, which is insufficient for network optimization. GOBLIN is a very powerful tool for solving network problems. It is a library of efficient codes for graph and network problems. Initially developed under Linux and later transferred to Windows environment. It has also good graphical user interface, but the main purpose is to implement various methods and algorithms, providing source code and excellent and detailed documentation. What LEDA offers is a complete collection of well-implemented data structures and types. Particularly useful is the graph type, which supports all the basic operations one needs in an intelligent way, although this generality comes at some cost in size and speed over handcrafted implementations. A small but useful library of graph algorithms is included. The Win Quantitative System for Business is an interactive decision support system which offers an array of powerful tools to help managers

* This paper is partially supported by the National Scientific Fund of the Ministry of Education and Science, contract No И1401/04 “Interactive Algorithms and Software Systems Supporing Multiciteria Decision Making”.

solve problems and make successful business decisions. Data of the graph models may be entered structured in the form of an adjacency matrix.

Structured architecture of a Graph and Network OPTImization software package (GNOPTI) is proposed in this paper, which may be of use both for educational purposes and for solving small and middle size real problems.

The purpose is twofold – to present an intuitively acceptable graphical environment for network optimization problems and also – to propose a tool for solving medium size real problems, which can be perspicuously formulated as network models. There is a lot of theoretical work and software since the early fifties of the past century till nowadays in the area of network optimization, e.g. [1].

The pure graphical approach is not convenient for solving network optimization problems on medium size and large networks. So the proposed package contains also tools for solving real problems specified in a list form. The exemplary realization is written in Visual C^{++} programming language for a WINDOWS 9xx and later platforms.

Some very short notes of the theoretical aspect for notations are given in Section 2. Section 3 briefly describes the architecture and modularity of the product as well as an outlook of the user interface. Section 4 discusses briefly the network optimization algorithms and their implementation in the package.

2. Preliminaries

2.1. Notations

The network optimization is one of the most developed areas in operations research [1, 2]. It is a powerful tool for solving satisfactorily a lot of arising problems in real world situations and thus any decision maker should have an idea of the network models and the methods for solving problems on them. Although from a theoretical point of view the network problems can be solved by the standard linear programming methods, the special structure of the problems made possible the development of specialized methods and algorithms which are much more efficient in time and memory than the linear programming approach.

Brief notational conventions follow for the sake of better readability.

Graph $\boldsymbol{G}$ is a structure (or set) defined by $\boldsymbol{G} = (V, E)$, where V is a finite set of vertices (nodes) indexed by natural numbers from 1 to n; E is a finite set of pairs of vertices, belonging to V, named edges, $|E| = m$. If the pairs are unordered, i.e. if $i \in V$, $j \in V$, then the pair (i, j) belongs to E, as well as the pair (j, i), the graph is undirected. If the pair (i, j) is ordered, then the nodes are defined as starting from i and ending into j. The pair (j, i) does not belong to E. In this case the graph is called directed or digraph. The directed edges are usually named arcs. Evidently an undirected graph can be easily transformed into a directed one by replacing the edge by a pair of opposite direction arcs.

A path is a sequence of edges (i_1, i_2), (i_2, i_3), ..., (i_{k-1}, i_k) belonging to E, which connect two nodes i_1, i_k belonging to V.

A graph is connected if a path exists between any two vertices. Otherwise the graph is disconnected.

If one or more parameters are defined on the arcs of a graph G, then this graph is called a network. Usually those are weight, capacity, gain, cost, etc. When a real function, called flow is defined on the graph set of arcs and a function of the flow value on the arcs should be optimized under given constraints, the problem is called a network optimization problem.

2.2. Representation of graphs and networks

Graphs and networks are represented in formal mathematical aspect either by adjacency or by incidence matrices.

The adjacency matrix is a $n \times n$ matrix. Each non-zero element of the adjacency matrix indicates that an edge exists between the vertices of the respective row and column indices. If the arc has parameters (weight, capacity, cost, etc.) the matrix element may be a record, containing their values. The adjacency matrix of an undirected graph is symmetric, while for directed graphs it is not.

The incidence matrix is a $n \times m$ matrix. In this matrix the element (i, j) is 1 if i is the starting node of the arc (i, j), and the element (j, i) is -1.

These matrices are very important, since by matrix operations on them many other matrices, describing different properties of the graphs may be easily found – e.g. cycles, a spanning tree, etc.

For connected graphs normally $m \geq n$. If $m = n^2$ the graph is called a fully connected graph.The matrices in real applications are sparse and if stored they occupy too much external storage. For instance, a 100 nodes graph if stored by the adjacency matrix requires 10 000 records, most of which zeroes. The case with the incidence matrix is even worse, since m is greater than n and besides this the specifying of parameters is more difficult.

The computer representation is somewhat more complicated, depending on the size and density of the matrices. The node connection list is proved to be the best way for external storage of the network because both the adjacency and incidence matrices may be easily reconstructed by it and hence, when needed – all other matrices – cuts, cycles etc.

This action is of complexity $O(n)$ both for time and memory [1].

The connection list looks like this:

A network of n nodes and of n rows is considered. Each element of the row i contains a record of the following structure $[(j), p_{(i,j)}(1)\ldots p_{(i,j)}(k)]$ where j is the node, connected by an arc emerging from i and ending in j and $p_{(i,j)}(.)$ are the arc parameters.

If the list presentation is available any matrix may be reconstructed.

DIMACS [8] is a very important format for representation of optimization problems and especially for networks. It is accepted as a standard for research and scientific investigation. It is a pure ASCII textual format and as so explicit preprocessing is necessary for conversion to internal representation of data. For network optimization a line of strict format is needed for problem description and for each node and arc. Being inherited from the main frames, it is not convenient for common PC users who are accustomed to user friendly interface.

3. Architecture and modularity

The software package GNOPTI is designed to solve single and multicriteria network optimization problems.

The single criterion network optimization is considered well formalized. This is not true concerning the multicriteria network optimization. Nonetheless new and more efficient methods and algorithms are proposed even for single criterion problems, concerning the huge increase of the networks size. In the first release of GNOPTI, only single criterion network optimization is included. The intention of the following release is to include multicriteria problems as well.

The purposes set in the design were as follows:

• a comprehensive graphical interface, mainly for educational goals;

• a possibility for defining on line parameters on nodes and arcs for solving different types of network optimization problems;

• a possibility of solving medium size problems in non-graphical mode;

• to produce a modular structure, in which solvers may be added for concrete purposes.

The purpose was also to make the interface as close as possible to the standard Win interface concerning the subject area.

The following architecture is proposed for the package:

Internal representation of networks – not strongly dependent on Windows rules,

User interface level – strongly dependent on Windows standards,

Solvers – independent on the operating system, written as self standing programs.

3.1. Internal representation of the network

For the purpose of saving space on the external carrier and for convenient presentation to the solvers, the Network after being entered or modified is COMPILED to create an object, which may be easily processed later on. The compilation also eliminates inconsistencies in the input data of the network.

Generally speaking the internal representation is a linked list, containing all the information of the network structure and the parameters of nodes and arcs. Any necessary representation may generated from it, specific for the needs of a given algorithm. Using the abilities of the object oriented Visual C^{++} by creating abstract classes like NET, NODE, ARC, etc. objects may be interactively defined, containing the methods and data, necessary for the concrete problem, not changing the code. E.g. different number of parameters may be specified for arcs, which are necessary for a concrete solver. The same is true for the nodes parameters.

In the case discussed, NET is the most abstract class. Exemplary C^{++} code of the class description is shown below.

```
#ifndef NET
#define NET
#include <iostream>
//////////////////////////////////////////////////////
//  Class Net represent common interface that must   //
//  inherit all rapresentations of network.                   //
//////////////////////////////////////////////////////
class Net
{
public:
    // add node to structure
    virtual void add_node() = 0;
    // remove node-th node
    virtual void del_node(unsigned int node) = 0;
    // set iterator to node-th node
    virtual void set_pos(unsigned int node) = 0;
    // get next child of node (set by set_pos) and store it number
    // to node, and arc’s number to arc. Increment iterator to next child.
    virtual bool next(unsigned int& node, unsigned int& arc) = 0;
    // random access ( like array ); return index of arc from i to j.
    // If there is no arc then return null (often 0).
    virtual unsigned int& operator() (unsigned int i, unsigned int j) = 0;
    virtual unsigned int operator() (unsigned int i, unsigned int j) const = 0;
    // destroy all data from structure
    virtual void destroy() = 0;
    // constructor (default nodes = 0; default null = 0);
    Net() {nodes = 0; null = 0;};
    // return number of nodes
    unsigned int nodes_count() {return nodes;};
    // print network information
    void print()
    {
            unsigned int node, arc;
            for ( unsigned int i = 0; i < nodes; i++ )
            {
                    set_pos(i);
                    while ( next( node, arc ) )
                    {
                         std::cout << i << “\t” << node << “\t” << arc << “\n”;
                    }
            };
    };
protected:
    unsigned int nodes;
    unsigned int null;
};
#endif
```

There are more classes, describing nodes and arcs, visual representation, and file manipulation, etc. mentioned in the NET header above.

3.2. User interface. Specifying and editing a network and its parameters

A new network may be specified either in an interactive graphical mode – most appropriate for educational purposes, or in a list mode – appropriate for networks with many nodes and arcs, which cannot be displayed on the working panel.

A network created in a graphical mode is shown in Fig.1. Nodes and arcs are created by clicking the mouse on the drawing canvas in a node editing and an arc editing mode respectively. Switching between these modes is done either from the main menu or (easier) from the graph drawing bar where the small circle (nodes) or the arrow (arcs) should be selected.

In the node editing mode, the user positions the mouse pointer on the canvas and left clicks. A new node is added and numbered.

In arc editing mode, the user clicks on the starting node and then on the ending node. An arc is created, numbered and displayed.

The network may exceed the visible area of the canvas. The window can be moved by the scroll bars.

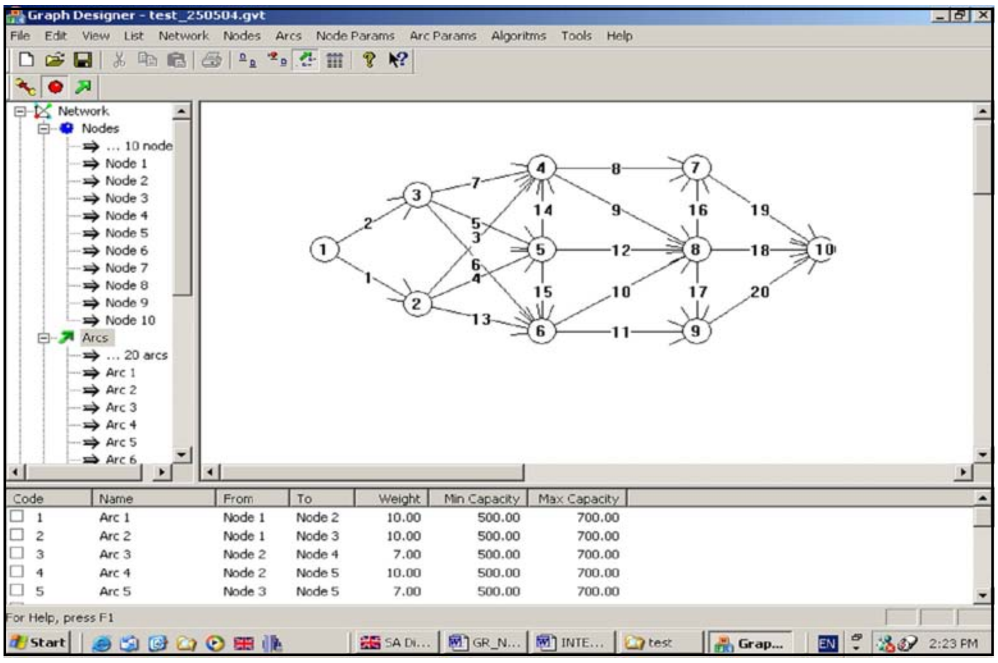
The graphical appearance of the network may be tailored by dragging nodes on the canvas. Then all incident arcs are also automatically adjusted.

In the left window a tree-like view of the nodes and arcs is seen in the sequence of their entering.

Parameters on the sets of nodes and/or arcs are specified in two steps. From the Node Params or Arc Params menu it may be selected which parameters are active. The respective structures are created and assigned default values. Then by clicking on the arc in the left window a series of dialog boxes are displayed for specifying the real values.

From the LIST menu an item may be selected pointing what type of the list is to be shown about the network – network, arcs, or nodes. When the list is displayed by selecting a given element, the parameters values may be copied to other elements. E.g. if Arcs view is selected, the parameters of the selected arc may be copied to other arcs.

After entering 10 nodes, 20 arcs and 3 arc parameters, the screen is as in the picture below. The min and max capacities are default, since they are of no interest for the example.

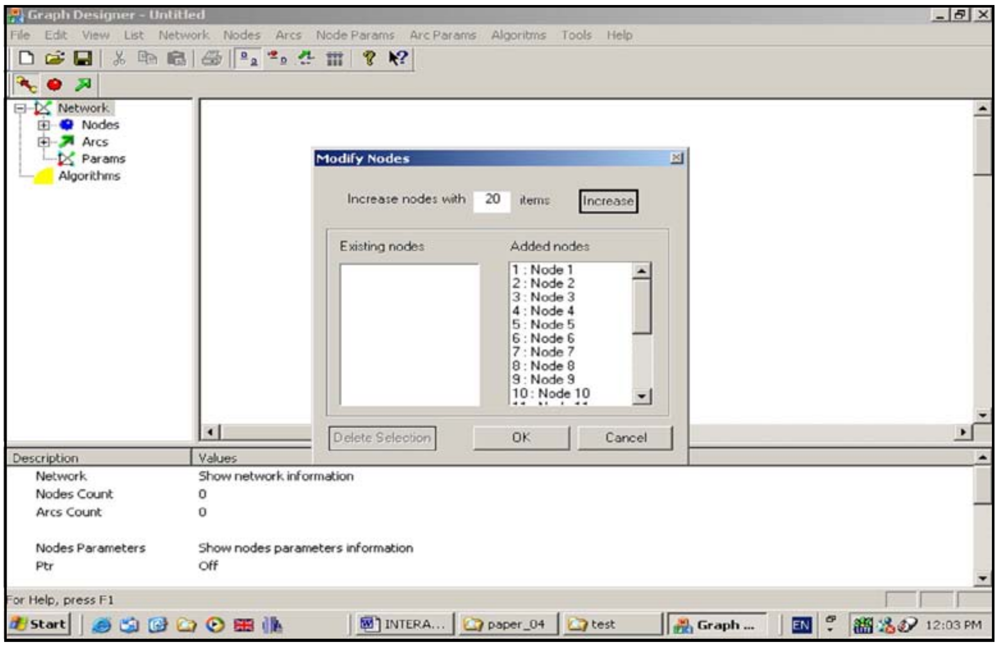


Code	Name	From	To	Weight	Min Capacity	Max Capacity
☐ 1	Arc 1	Node 1	Node 2	10.00	500.00	700.00
☐ 2	Arc 2	Node 1	Node 3	10.00	500.00	700.00
☐ 3	Arc 3	Node 2	Node 4	7.00	500.00	700.00
☐ 4	Arc 4	Node 2	Node 5	10.00	500.00	700.00
☐ 5	Arc 5	Node 3	Node 5	7.00	500.00	700.00

Fig. 1

There are a lot of other possibilities to set parameters to arcs and/or nodes, their values, etc. All this is a subject of the Users' Guide.

There is also another way to specify networks of many nodes and arcs, which is appropriate for solving medium size networks, which are inconvenient to enter in graphical mode. Let us consider a network of 20 nodes and 100 arcs. Then the number of nodes is entered by clicking on "Network" on the main bar and then selecting "Modify nodes". A window appears, asking how many nodes to add. When the number is entered and the right arrow in the middle of the screen is clicked, the nodes are automatically added. Then from the item "Modify arcs", by selecting the starting node in the right window and then by selecting the list of end nodes in the left window, the new arcs of the network may be specified. In the Figs. 2 and 3 are shown the appearance of the respective windows.

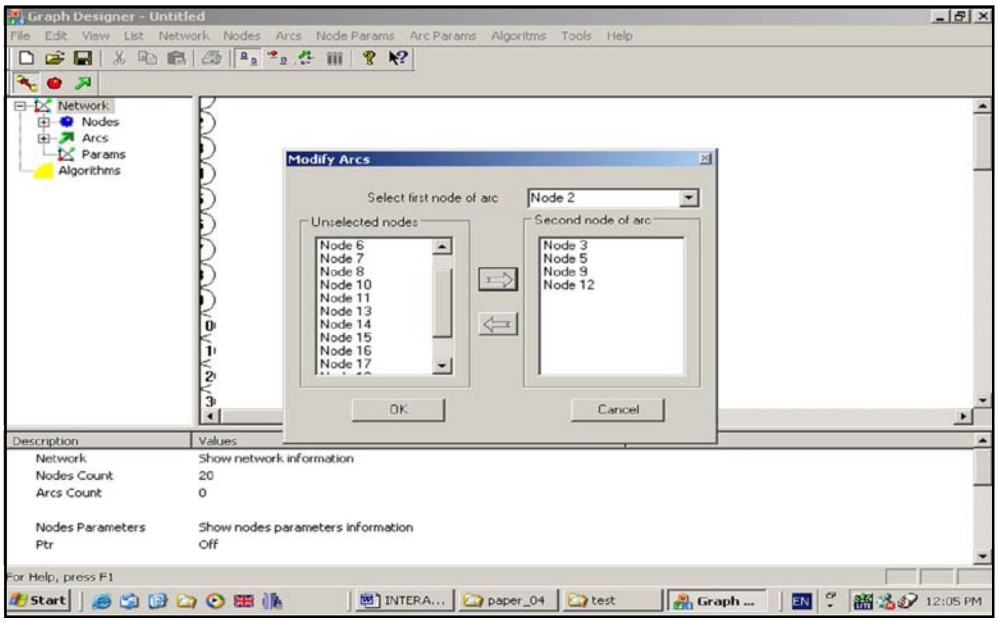


Increase nodes with 20 items

Existing nodes	Added nodes
	1 : Node 1
	2 : Node 2
	3 : Node 3
	4 : Node 4
	5 : Node 5
	6 : Node 6
	7 : Node 7
	8 : Node 8
	9 : Node 9
	10 : Node 10

Description	Values
Network	Show network information
Nodes Count	0
Arcs Count	0
Nodes Parameters	Show nodes parameters information
Ptr	Off

Fig. 2



Select first node of arc: Node 2

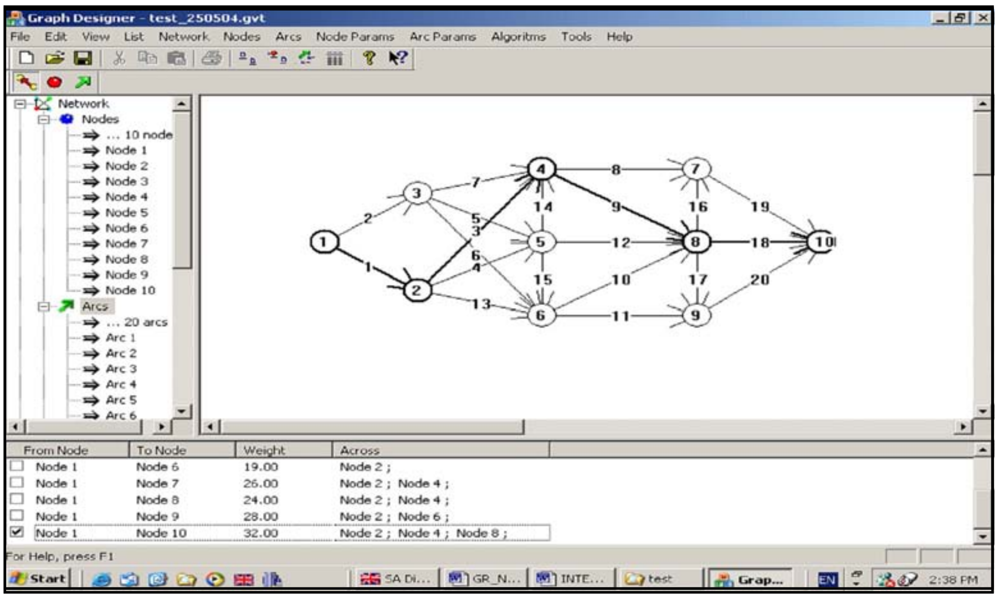
Unselected nodes	Second node of arc
Node 6	Node 3
Node 7	Node 5
Node 8	Node 9
Node 10	Node 12
Node 11	
Node 13	
Node 14	
Node 15	
Node 16	
Node 17	

Description	Values
Network	Show network information
Nodes Count	20
Arcs Count	0
Nodes Parameters	Show nodes parameters information
Ptr	Off

Fig. 3

The nodes appear on the left side of the drawing canvas. They may be dragged and put on the canvas to have a visual representation for a part of the network.

When solving a problem from Algorithms item of the main menu, an algorithm is selected and then in the dialog windows the necessary initial information is supplied. In the Fig. 4, the solution of the shortest path problem by Dijktra's algorithm is shown for the exemplary network.



From Node	To Node	Weight	Across
☐ Node 1	Node 6	19.00	Node 2 ;
☐ Node 1	Node 7	26.00	Node 2 ; Node 4 ;
☐ Node 1	Node 8	24.00	Node 2 ; Node 4 ;
☐ Node 1	Node 9	28.00	Node 2 ; Node 6 ;
☑ Node 1	Node 10	32.00	Node 2 ; Node 4 ; Node 8 ;

Fig. 4

These are just illustrations of the interface windows. The technical details are a subject of the Users' Guide.

Once entered, the network may be modified and parameters edited for new problems.

4. Solvers

The solvers are not necessarily apriori defined. Newly developed programs may be included realizing new algorithms and/or removing some existing ones. There is no need to redesign the interface in connection with these changes. The new programs should of course comply with the internal representation of the network. This is not a problem, since the solvers are defined as classes where inter-object connections are implicit. When implementing a new algorithm, the user should write a header for the new class and then - the necessary executable code.

This is shown in the small header of Dijkstra's solver, so that the necessary data are easily imported from the network description structures:

```
#ifndef  DIJKSTRA
#define DIJKSTRA
#include "network.h"
#include <iostream>
```

```
const unsigned MAX_DIM = 100;
const unsigned MAX_VALUE = 1000000;
const unsigned NO_PARENT = –1;
class Dijkstra
{
public:
   Dijkstra(Network* n);
   ~Dijkstra();
   void evaluate(unsigned node);
private:
   void print(unsigned node);
   void print_path(unsigned start, unsigned end);
   Network* net;
   unsigned nodes;
   WEIGHT_TYPE* distance;
   unsigned* parent;
   short* considered;
};
#endif
```

The most important types of network optimization intended to be included in the package are:

- the shortest path between a pair of nodes;
- the shortest paths between all pairs of nodes;
- *k* shortest paths between a pair of nodes;
- the minimum spanning tree;
- flow optimization (max or min);
- a traveling salesman;
- a Chinese postman;
- an optimal flow in a network with gains and losses.

The next step intended is to include multicriteria network flow problems in the package. The flow function is defined on the network arcs, subject to specific constraints. The goal of the network flow optimization problem is to optimize an objective function finding the optimal linear combination of the flow values on the arcs. Usually some parameters, e.g. weights, costs, etc. are coefficients in this linear combination. If there is more than one objective function, then the problem is said to be multicriterial. Although some network flow problems are implicitly multicriterial, e.g. min cost max flow, multicommoditty flow, a very simple example, which cannot be covered by them, is when two flows of different weights and costs on the arcs are considered and one of them is to be maximized, and the other one – minimized with respect to one or both of the parameters.

5. Conclusion

The Graph and Network Optimization package proposed, which is designed as an open architecture, may be of use for educational purposes, for solving real problems, and for researchers experimenting their own algorithms. The idea of the interactive specifying of the parameters on arcs and nodes, decreases the efforts for entering new networks of a similar structure. The compilation of the network helps the preserving in a compact form of the network entered. The definition of solvers as classes is convenient since the researcher may concentrate on the methods and algorithms that he/she would like to implement, not sparing efforts on the network representation.

References

1. A h u j a, R., T. M a g n a n t i, J. O r l i n. Network Flows. Englewood Cliffs NJ, Prentice Hall, 1993.
2. J u n g n i c k e l, D. Graphs, Networks and Algorithms. Springer Verlag, 1999.
3. D i l l w o r t h, D., C. R. C o u l l a r d, J. H. O w e n. Graph&related Data Structures User's Guide. Technical Report TR-96.09, Dept. of Industrial Eng. and Man. Sci., Northwest Univ., 1996.
4. GOBLIN – a Library for Graph Matching and Network Programming Problems. Ref. Manual by Christian Fremuth-Paeger, Sept. 2000-Sept.2002. Univ. Augsburg, Inst. Fur Mathematik.
5. P e t c h e n k i n e, V i t a l i. GRAPH INTERFACE GRIN. **http://www.geocities.com/pechv_ru/**
6. M e h l h o r n, K., S. N a h e r. LEDA, a platform for combinatorial and geometric computing. – Communications of the ACM, **38**, 1995, 96-102.
7. Y i h-L o n g C h a n g, K i r a n D e s a l. WinQSB, Version 2.0. ISBN: 0-471-40672-4, Wiley, 2002.
8. Network Flows and Matching: First DIMACS Implementation Challenge. (David S. Johnson and Catherine C., Eds.). Volume 12. McGeoch; AMS, Providence, RI, 1993.

Программный пакет для оптимизации на графах и сетях

Станислав Дрангажов, Васил Василев, Николай Димитров, Мариана Джелатова

Институт информационных технологий, 1113 София

(Р е з ю м е)

Представлен программный пакет для оптимизации на графах и сетях, названный GNOPTI. Пакет предлагает графический интерфейс для цели образования, а также и возможности для решения разнообразных реальных задач оптимизации в сетях. Сделан обзор самых распространенных систем для сетевой оптимизации. Подробно обсуждены архитектура, потребительский интерфейс и программные соображения.