AlgoDeGrafos: An Application to Assist in Course Lectures on Graph Theory

Denis S. Silveira, Valdir A. Melo, Paulo O. Boaventura Netto

Program of Production Engineering – COPPE/UFRJ
P.O. Box 68.511 – CEP 21.941-972
Rio de Janeiro – RJ - Brazil
{denis, vmelo, boaventu}@pep.ufrj.br

Abstract

This paper describes an application of the object-oriented paradigm to the construction of *AlgoDeGrafos,* a graph-theoretical system enabling the use of graph algorithms. This system is freely available for academic purposes and can be used as a support in courses on graph theory algorithms.

Keywords: graphs, graph algorithms, object-oriented paradigm.

1. INTRODUCTION

The growing importance of graph theory in present times is linked to the fact that it allows for the development of models in several knowledge areas, such as computer science, mathematics, genetics, chemistry, telecommunications and engineering, in order to furnish representations of real-world problems.

Learning about applied graph theory involves the understanding and use of its more traditional algorithms, with this being an appropriate time for the lecturer to build suitable examples and, frequently, to introduce his/her students to a small, real-world application. Generally, this process involves the implementation of specific algorithms and/or a search for appropriate codes, thus requiring previous programming knowledge which may not necessarily be uniformly present within a given group of students. Students would consequently be forced to do additional work outside the course focus, which may constitute yet another hurdle for the learning process.

Another relevant question concerns students' first contact with graph algorithms and the level of abstraction they require. When seeking to understand steps in the process, it is convenient to mentally visualize the algorithm's execution, but available classroom time may be insufficient for this activity, even with the help of graph schemes. This could prove to be very discouraging for the students. Such a situation calls for the use of new methodologies in the educational process, in order to enhance student interest. Pimentel [11] discusses ways for the intelligent and creative use of the computer as an aid in the teaching/learning process, allowing faster and easier transmission and assimilation of information and knowledge.

Our contribution to these activities is the *AlgoDeGrafos* tool, a free didactic system designed to help undergraduate and graduate students dealing with graph theory. Through its use, beginners can more easily acquire a deeper understanding of graph structures, work with the theory's concepts in a more intuitive manner and verify its use through various examples, whereas the more traditional means of introduction via lectures is largely theoretical. But the chief contribution we seek to make is providing interactivity, manipulation and context control by the student, resources which, according to Gouveia [5] reinforce motivation, put the student more at ease and facilitate his/her understanding.

Current thinking/practice recognizes three ways to conduct a teaching/learning process, those being via mutual presence, semi-presence and remote processes. Mutual presence is the conventional process, where lecturer and students have scheduled meetings in fixed places, with linear presentations on given subjects. Everyone is then supposed to learn following the pace and order laid out by the lecturer. The semi-presence process takes place partly in the classroom and partly outside of it, with the use of convenient tools, while the remote process is based on having the lecturer and students physically separated in space and time.

The tool we present in this work was developed with the chief intention of offering an aid to courses based on mutual presence, aiming to provide more dynamic classroom interaction, augment students' interest and participation and generate circumstances in which students can resolve some of their doubts in the absence of the lecturer. The *AlgoDeGrafos* allows the execution of a number of graph algorithms, offering a user-friendly interface which has both Help documentation and an interactive assistant. Through these, a user can easily find the successive commands needed to attain his/her objectives. Graph data can be entered both by file reading and interactively. The system comes with some pre-installed examples to help with visualization and the application of the algorithms. New graphs can then be entered and the conveniently ready processes applied to them.

AlgoDeGrafos can be used by a student wanting to check the solution given by his/her model and implementations and to check results of exercises solved by hand. For the lecturer, it is an auxiliary tool for quickly checking the results of tutorials, work done outside of class and tests, and also to check solutions given by the students.

Section 2 shows a comparison between *AlgoDeGrafos* and other tools available on the Internet. Section 3 describes the problems and the algorithms provided for their solution, Section 4 is devoted to the tool's design, discussed through an outline of its class diagram. Lastly, Section 5 shows a practical application of the system, following which we present our final considerations.

2. SIMILAR TOOLS

There are other graph-theoretical systems available on the Internet. We see the use of *AlgoDeGrafos* as self-evident, given the characteristics of other systems, in part due to inherent restrictions in the latter, and because many are not open-source, thus involving a commitment of financial resources by the student or by the institution.

We can find programs like [4] and [6], devoted to specific algorithms or problems. Others have plenty of algorithms and more sophisticated elaboration but, as noted, frequently are not free and do not present an intuitive interface. Other cases, like [12] and [13], provide only functions to be included in user-developed programs, which in part leave the user again confronting the abovementioned difficulties.

AlgoDeGrafos allows for the elaboration, application and visualization of certain graph theory problems and also the solution of practical problems of small order (up to 50 vertices). Nevertheless, there is no restriction on one's capacity to edit and visualize greater order graphs.

Through Internet searches, we have chosen for comparison two generic tools and a dedicated program, as follows:

- GRIN - *Graph Interface*: Developed by Vitaly Pechenkin. This is presently one of the more complete tools for graph manipulation. Both free and licensed versions can be obtained at: http://www.geocities.com/pechv_ru/ [9];
- *Graph-Magics*: Developed by Dumitru Ciubatii. This tool allows the user to deal with various graph theory problems. Both free and licensed versions can be obtained at: http://www.graph-magics.com/ [2];
- DSPA - *Dijkstra's Shortest Path Algorithm*: Developed by Harvey J. Greenbeng. This is a tool for solving the shortest path problem. The site is http://carbon.cudenver.edu/~hgreenbe/sessions/ dijkstra/DijkstraApplet.html [4].

GRIN has an efficient graphical interface to create and visualize graphs and it offers several graph algorithms. Nevertheless, the full system is not free, and the free version has a number of limitations.

Graph-Magics also offers efficient instruments to build graphs. For instance, vertices and edges can be added with just one click; editing weights and capacities can be done directly on the graph or one can use a table. The system description [2] claims that it can be used with graphs having more than 1000 vertices. We have not checked this potential, as we used the free download version, which remains available for only 30 days.

DSPA is an *Applet Java* designed for solving the shortest-path problem. It allows the introduction of vertices and edges and also weight edition directly on the graph scheme, directly on the Internet, but its operating capacity is limited as it solves only the shortest-path problem through the Dijkstra algorithm.

For comparison, we considered ease of use, I/O resources, implemented algorithms and also interface, limitations, restrictions, help and installation. Table 1 below shows the main characteristics of each tool (the evaluated GRIN version was the free one).

Table 1: Tool Comparison

Feature	**GRIN**	**Graph-Magics**	**DSPA**	***AlgoDeGrafos***
Read and save the graph in files	yes	yes	no	yes
Graphic data input	yes	yes	yes	yes
Graphic display of results	yes	yes	yes	yes
Step-by-step execution	yes	yes	no	no
Multiplatform	no	no	yes	no
Completely free	no	no	yes	yes
Allows for directed edges (arcs)	no	no	yes	yes
Allows for undirected edges	yes	yes	no	yes
Help topics	yes	yes	yes	yes
Edge breaking-point allowed	no	no	no	yes
Algorithms: minimal spanning tree	yes	yes	no	yes
Algorithms: vertex-coloring	yes	yes	no	yes
Algorithms: shortest path	yes	yes	yes	yes
Algorithms: maximum flow	yes	yes	no	yes
Algorithms: minimum spanning arborescence	no	no	no	yes
Algorithms: levels of a circuitless graph	no	no	no	yes
Algorithms: PERT method	no	no	no	yes
Algorithms: Chinese postman problem	no	yes	no	no
Maximum graph order for algorithm execution	40	1000	20	50

3. THE ALGORITHMS IMPLEMENTED

The choice of algorithms for the present *AlgoDeGrafos* version took into account the most generally known problems in graph theory, [1]. For the majority of them, more than one algorithm is presented. In what follows, we offer a brief discussion of the problems and algorithms considered.

3.1 The Shortest Path Problem

There are two basic types of shortest path problems: those which find minimum paths from a given vertex and those which allow the determination of minimum paths between every vertex pair in the graph. Naturally, one can always apply n times an algorithm designed for a first-type problem to solve a second-type one. The problem exists both in directed and undirected graphs. The implemented algorithms for first-type problems are *Dijkstra, Dantzig* e *Bellman-Ford* ones; for second-type problems, the system offers the *Floyd* algorithm.

3.2 The Vertex Coloring Problem

Coloring problems deal with situations such as the construction of courses timetables and working shifts. A vertex-coloring in a graph is the allocation of a color to every vertex, in such manner that no pair of adjacent vertices has the same color. The lesser number of colors that fulfill this constraint for a given graph is its chromatic number. We present two heuristics for the approximate solution of this problem, that of *class coloring* and that of *sequential coloring* and, to obtain the exact chromatic number, the *Zykov* algorithm.

3.3 The Maximum Flow Problem

A flow in a graph is here considered as the traffic of discrete units along the graph links, following some capacity and equilibrium constraints. Flow problems are related to applications such as road traffic, cargo transportation and paper and decision flows in organizations. We consider an origin and a destination for a maximum flow to be determined, its value corresponding to the net minimum capacity among the graph cuts (the graph bottleneck). *AlgoDeGrafos* presents the *Ford-Fulkerson* and *DMKM* (from *Dinic*, *Malhotra et al*) [1] algorithms for this problem.

3.4 The Minimum Spanning Tree Problem

This is an optimal linking problem in undirected graphs which frequently are models of distribution networks where the cost of network elements is independent of the distance to a given key point (such as a distribution center). We want to find an edge subset which connects every vertex without creating cycles. Such a subset defines a *spanning tree* and our interest is to obtain one with minimum cost. We offer the *Prim* algorithm, designed for iterative vertex addition and *Kruskal* algorithm, which adds edges to the tree.

3.5 The Minimum Spanning Arborescence Problem

This problem corresponds to the search of a spanning arborescence in a directed graph which has the minimum cost among every arborescence having a given root. It differs from the previously cited problem by the existence of a key vertex (the *root*) and by the need to guarantee of a single path between the root and every other vertex. We provide the *Gondran-Minoux* algorithm.

3.6 The Level Partition in a Circuitless Graph

Finding the levels in a circuitless graph is an easily solved problem with an immediate application, which is to verify the correctness of a PERT model. The *Democron* algorithm is presented for this problem.

3.7 The PERT Method

The PERT method is a technique for sequencing the activities of some type of work or *project* that allows it to be divided into a number of *tasks*. The PERT algorithm allows one to determine the minimum time required to complete the project as is, and also to study resource reallocation in order to reduce execution time. It involves three phases, those of determining the earliest and the latest dates for each project event and, finally, the determination of a critical path, formed by those tasks that cannot undergo any delay, and which indicates the total time to be consumed by the project.

4. TOOL DESIGN

AlgoDeGrafos was built by using UML (*Unified Modeling Language*) [8], which is presently the standard for object-oriented model building, both in businesses and in academic work. The graphical modeling languages have been used for a long time in the software industry, owing to their abstraction level, higher than that of programming languages, allowing one to more easily think about the concepts involved in a project [3].

Following this path, the first step related to the project was to examine and identify the concepts pertinent to the problem by defining a metamodel representing a graph, that is, a pair $G = (V,E)$, where V is a finite set of vertices and E is an edge set, with each edge being a link within an unordered pair (v_i, v_j) of vertices.

The Figure 1 shows this metamodel through MOF (Meta-Object Facility). MOF is a standard defining an abstract language and a structure for the specification, construction and management of metamodels in a technology-independent way [7]. The constructions defined in MOF follow the object-orientation paradigm and define a basic set of elements: class, attribute, operation, binary association between classes and heritage. Like UML, MOF is a standard adopted by OMG (Object Management Group) which since then has been responsible for its development.

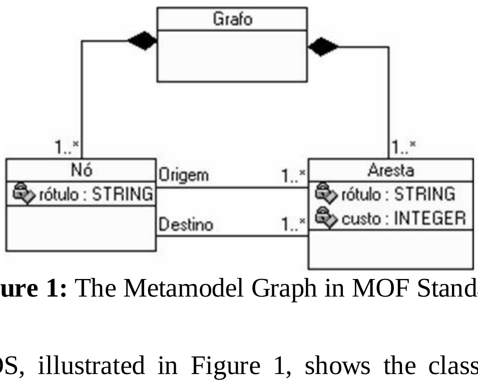


Figure 1: The Metamodel Graph in MOF Standard.

The metamodel GRAFOS, illustrated in Figure 1, shows the classes *Grafo*, *Nó* and *Aresta*. These classes are the kernel of the *AlgoDeGrafos* tool, offering a reutilization structure in the context of the project. Figure 2 below illustrates partially the tool dominium classes.

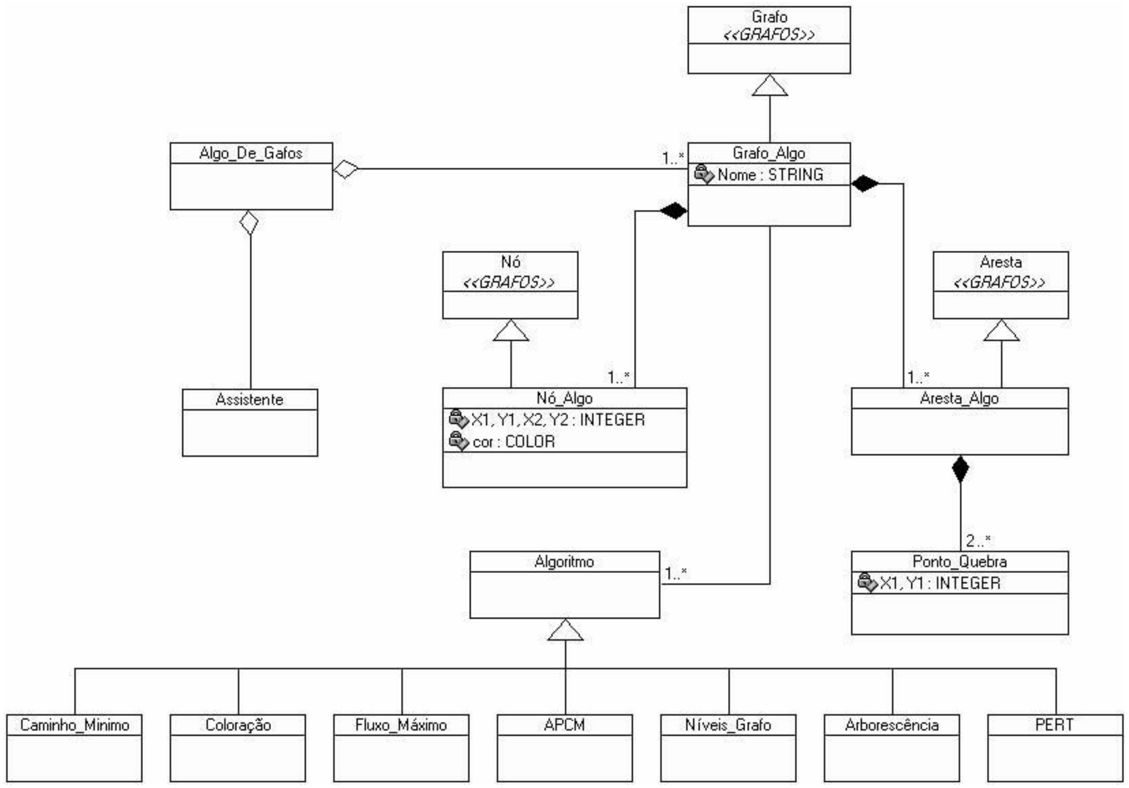


Figure 2: Partial Classes Diagram for *AlgoDeGrafos*.

In the figure, we can observe the *AlgoDeGrafos* class, which represents the main object, that is, the application itself. Starting with this object, all functionalities are delegated to the remaining objects. The classes *Grafo_Algo*, *Nó_Algo* and *Aresta_Algo* are respectively the specializations of the classes from the GRAFOS metamodel shown in Figure 1. That is, they customize these classes with properties and behavior pertinent to the tool. For instance, the *Nó_Algo* class inherits the property (attribute) label from *Nó* class. This attribute is pertinent to every application able to work with a graph vertex, independently of the application. Nevertheless, the attributes *X1*, *Y1*, *X2* and *Y2* are pertinent only for graphical applications, for instance, *AlgoDeGrafos*, where one needs to draw a vertex according to its coordinates (a specific point for the vertex in the graph). We have the same situation with the property *color*, which allows the user to associate any color to the vertex.

In the model, we can observe that the instances from *Aresta_Algo* class are composed of a minimum of two and a maximum of *n* instances from the objects of the *Ponto_Quebra* class. Each instance from this class represents a graphical point linking an edge to a vertex (two cases) or an edge break. The example shown in Figure 3, below, has five instances from the *Ponto_Quebra* class, two of them being pertinent to vertex associations and three to edge breaks.

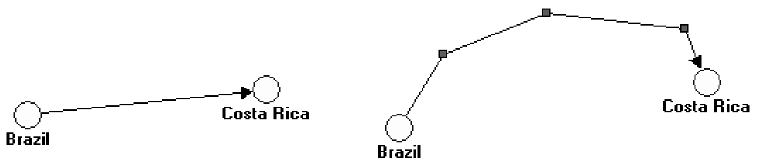


Figure 3: A Visual Example for Edge Breaking Point.

The *Algorithm* class is an abstract one and was used to join the properties and behaviors pertinent to every implemented algorithm within *AlgoDeGrafos*. There, the specializations are pertinent to the following problems:

- *Caminho_Mínimo (shortest path)*: implements the *Dijkstra*, *Dantzig*, *Bellman-Ford* and *Floyd* algorithms, pertinent to the shortest path problem;
- *Coloração (coloring):* implements the class and sequential coloring algorithms;
- *Fluxo_Máximo (maximum flow):* implements the *Ford-Fulkerson* and *DMKM* algorithms, pertinent to the maximum flow problem;
- *APCM (minimum cost spanning tree):* implements the *Prim* and *Kruskal* algorithms, both pertinent to the optimal interlinking problem in undirected graphs;

- *Arborescência (arborescence)*: implements *Gondran-Minoux* algorithms, pertinent to the problem of the optimal spanning arborescence with given root in a directed graph;
- *Níveis_Grafo (graph levels)*: implements the *Demoucron* algorithm, pertinent to the problem of finding the levels in a circuitless graph;
- *PERT*: implements the *PERT* algorithm, pertinent to the problem of minimizing execution time and allowing for revision of a project divisible into tasks.

Finally, the model presents the *Assistente* class. It is responsible for the instance of a visual assistant, which is an animated character whose objective is to improve communication with the user and attract his/her attention at given moments, aiming to emphasize information or presentations of recommendations. This type of communication mechanism has been used in a number of applications, such as electronic commerce, e-learning and help-desks.

5. A PRACTICAL APPLICATION

The allocation of security points (cabins and other installations adequately equipped) allows to effectively enhancing public security in situations demanding quick police intervention. This example is taken from a pilot study concerning part of the Leblon suburb in the city of Rio de Janeiro. The study considers the presence of at least one police vehicle ready for use as a consequence of a call or through decision of the policemen [10]. It is very convenient for a comparison between the use of the *AlgoDeGrafos* environment and other available resources because the original data base was not compatible with graph algorithms, a situation that required rather demanding adaptation work. Figure 4, below, shows part of the original model representation.

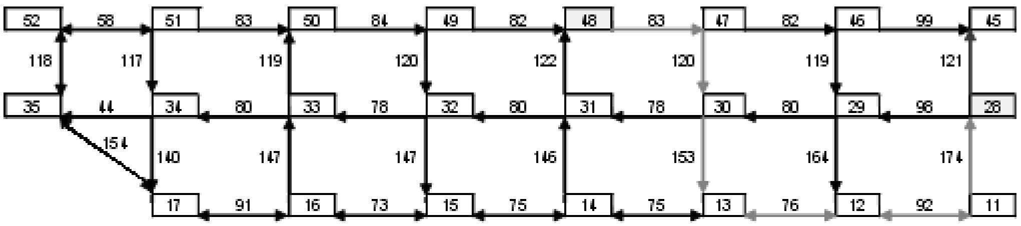


Figure 4: Partial Representation of the Original Model.

The Figure 5, below, shows the same content as represented in the *AlgoDeGrafos* environment.

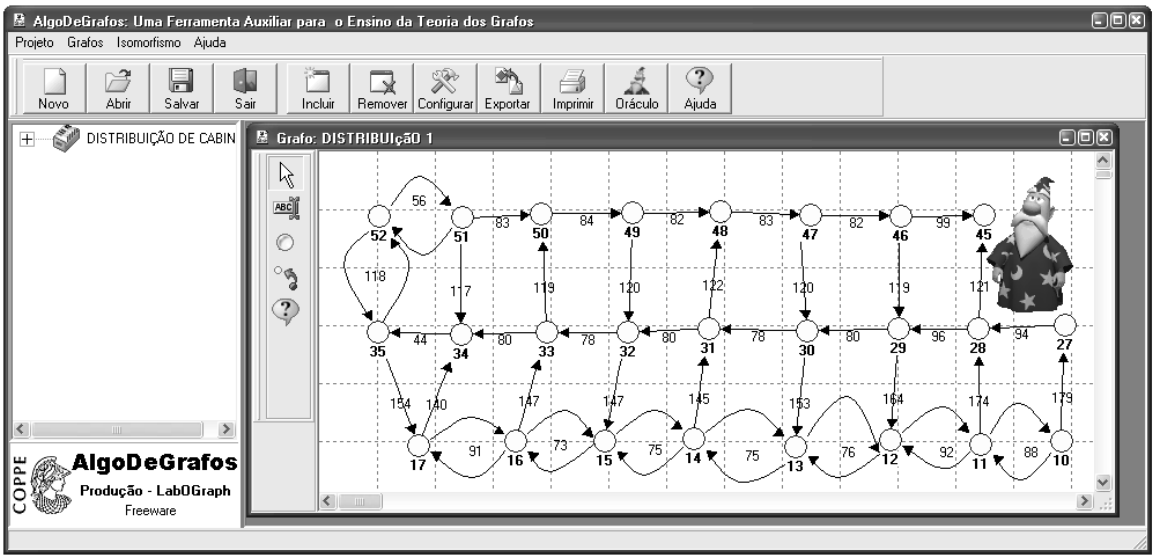


Figure 5: The Corresponding Graph in *AlgoDeGrafos*.

Another advantage is that a student can use a saved project to test other algorithms, provided that the graph is compatible with the type admitted by the algorithm. We can, for instance, use the graph in Figure 5 to apply *Dijkstra's* algorithm which, as we have seen, is a classic shortest-path algorithm. To access the algorithm menu, one needs only to right-click the graph's free area and select the desired option. See Figure 6, below.

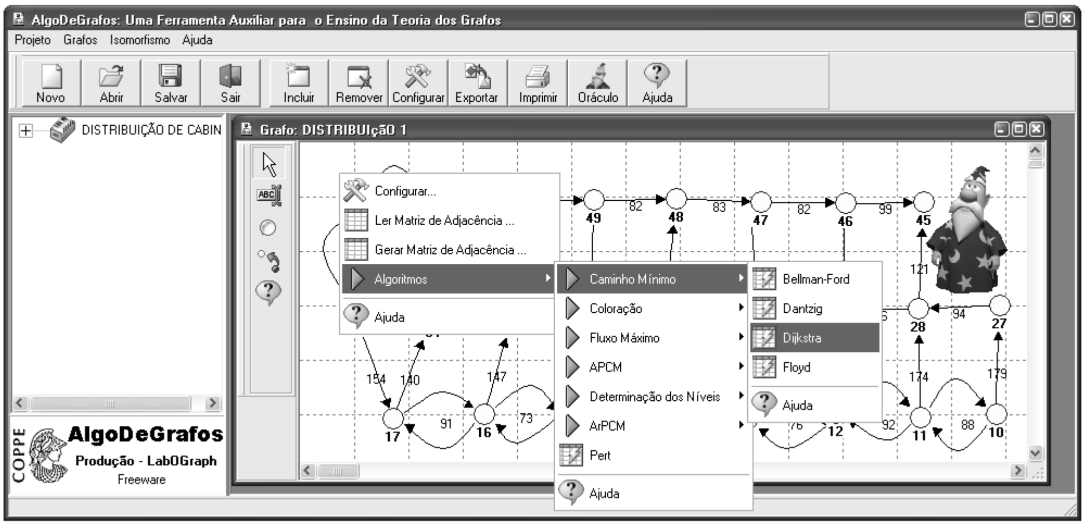
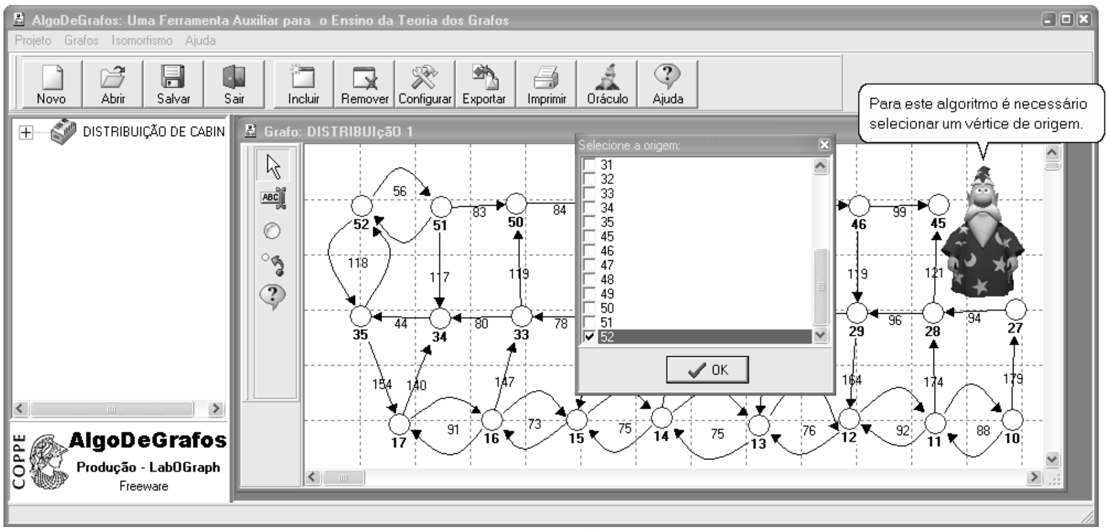

AlgoDeGrafos: Uma Ferramenta Auxiliar para o Ensino da Teoria dos Grafos
Projeto Grafos Isomorfismo Ajuda
Novo Abrir Salvar Sair Incluir Remover Configurar Exportar Imprimir Oráculo Ajuda
DISTRIBUIÇÃO DE CABIN
Grafo: DISTRIBUIçãO 1
Configurar...
Ler Matriz de Adjacência ...
Gerar Matriz de Adjacência ...
Algoritmos
Ajuda
Caminho Mínimo
Coloração
Fluxo Máximo
APCM
Determinação dos Níveis
ArPCM
Pert
Ajuda
Bellman-Ford
Dantzig
Dijkstra
Floyd
Ajuda
AlgoDeGrafos
Produção - LabOGraph
Freeware
COPPE

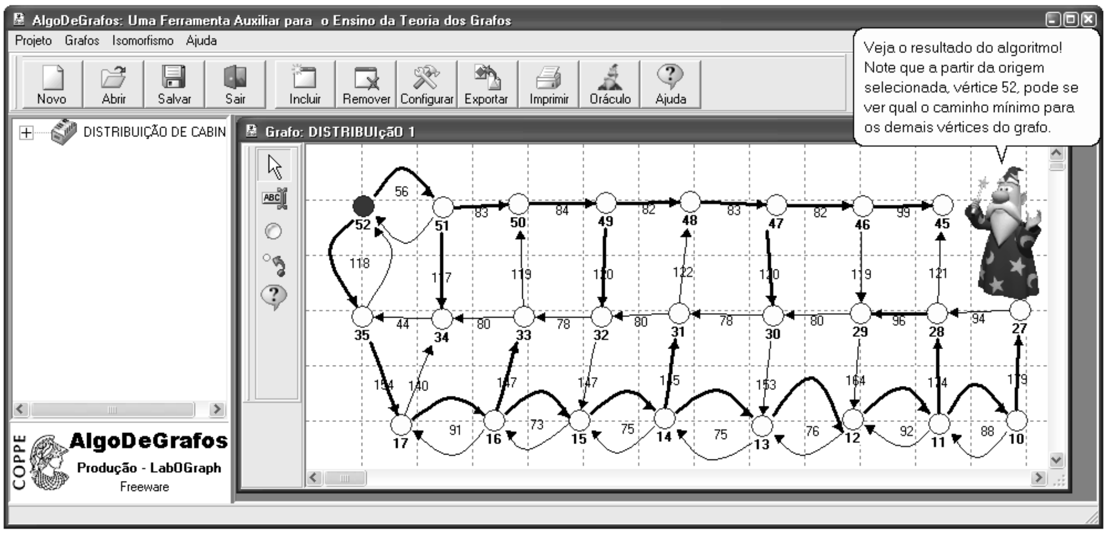
Figure 6: Selecting the Desired Option for Algorithm Execution

Let us recall that the shortest-path problem exists both in directed and undirected graphs. The *Dijkstra* algorithm is formulated for the first case and, because of that, it requires a starting point to begin. At this moment the *Merlin* figure, the *AlgoDeGrafos* assistant, asks for the selection of such an origin, using voice and written resources: “For this algorithm, you have to select an origin vertex”. See Figures 7 and 8, below.


Para este algoritmo é necessário selecionar um vértice de origem.

AlgoDeGrafos: Uma Ferramenta Auxiliar para o Ensino da Teoria dos Grafos
Projeto Grafos Isomorfismo Ajuda
Novo Abrir Salvar Sair Incluir Remover Configurar Exportar Imprimir Oráculo Ajuda
DISTRIBUIÇÃO DE CABIN
Grafo: DISTRIBUIçãO 1
Selecione a origem:
31 32 33 34 35 45 46 47 48 49 50 51 52
OK
AlgoDeGrafos
Produção - LabOGraph
Freeware
COPPE

Figure 7: Selecting an Origin for *Dijkstra’s* Algorithm.


Veja o resultado do algoritmo! Note que a partir da origem selecionada, vértice 52, pode se ver qual o caminho mínimo para os demais vértices do grafo.

AlgoDeGrafos: Uma Ferramenta Auxiliar para o Ensino da Teoria dos Grafos
Projeto Grafos Isomorfismo Ajuda
Novo Abrir Salvar Sair Incluir Remover Configurar Exportar Imprimir Oráculo Ajuda
DISTRIBUIÇÃO DE CABIN
Grafo: DISTRIBUIçãO 1
AlgoDeGrafos
Produção - LabOGraph
Freeware
COPPE

Figure 8: The Application of *Dijkstra’s* Algorithm with Vertex 52 as Origin.

We can see that the origin is indicated in red and the shortest paths starting from it are presented in bold lines. The assistant reinforces this information: “Look at the result given by the algorithm! Observe that we can start from the selected origin, vertex 52, and see the shortest paths to the remaining vertices.”

6. FINAL CONSIDERATIONS

This work has presented the *AlgoDeGrafos* application, a free software with a number of capacities suitable to be used in instructional lectures on graph theory, such as the introduction of examples both by drawing and by file reading and the use of several algorithms among those most often used with graph applications. Through its use, students beginning studies in graph theory will be able to acquire a deeper perception of graph structures, work with its concepts in a more intuitive way and check their examples.

One main contribution of *AlgoDeGrafos* is the information on graph theory available through its Help feature, also presented using the animation and audio of the assistant, which offers excellent potential to improve student learning.

We are currently working to improve *AlgoDeGrafos* through the introduction of new functions such as the building of line graphs and the introduction of a heuristic to study graph isomorphism.

References

[1] Boaventura-Netto, P. O., *Grafos: Teoria, Modelos, Algoritmos*, 4th. Edition, Edgard Blücher, São Paulo, 2006.

[2] Ciubatii, D., *Graph-Magics*, Available at http://www.graph-magics.com/ - Accessed in November 2006.

[3] Fowler, M., *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, 3 th Edition, Pearson Education, Inc., 2004.

[4] Greenbeng, H. J., *Dijkstra's Shortest Path Algorithm*, Available at: http://carbon.cudenver.edu/~hgreenbe/sessions/dijkstra/DijkstraApplet.html -Accessed in November 2006.

[5] Gouveia, L., Camacho, M., *Criação de Espaços de Informação Interativos – Ambiente de aprendizagem para a cadeira de “Sistemas de Informação”*. III Simposio Investigação e Desenvolvimento de Software Educativo, 1998.

[6] McKay, B. D., *Nauty is a Program for Computing Automorphism Groups of Graphs and Digraphs.* Available at: http://cs.anu.edu.au/people/bdm/nauty/ Accessed in October 2006.

[7] OMG. Object Management Group – *MOF 1.4 specification*, at: http://www.omg.org/cgi¬bin/doc?formal/2002-04-03, 2002.

[8] OMG. Object Management Group – *Unified Modeling Language: Infrastructure, Version 2.0,* at: http://www.omg.org/docs/formal/05-07-05.pdf, 2006;

[9] Pechenkin, V., *GRIN - Graph Interface*, Available at: http://www.geocities.com/pechv_r1/ - Accessed in November 2006.

[10] Pereira, R., Garcia, L., Melo, V., Texeira, P., Boaventura-Netto, P. O. *Distribuição de Cabines de Segurança em parte do Bairro do Leblon na Cidade do Rio de Janeiro*, Simpósio Brasileiro de Pesquisa Operacional – SBPO 2004.

[11] Pimentel, C., *A Informática na Prática Interdisciplinar;* Anais do XIX Congresso Nacional da Sociedade Brasileira de Computação; pp. 919-925; Rio de Janeiro, 1999.

[12] Sangiorgi, U. B., *Rox: Um Framework Open Source para a Construção de Aplicações Baseadas em Grafos*. Available at: http://gnu.frb.br:8080/rox - Accessed in October 2006.

[13] Siek, J., Lee L. and Lumsdaine, A., *A C++ Library for Graph Algorithms*. Available at: http://www.boost.org/libs/graph/doc/table _of_contents.html - Accessed in April 2006.