A System to Evaluate and Understand Routing Algorithms

Mario Marcelo Berón
Univ. de San Luis
Argentina
mberon@unsl.edu.ar

Perdo Rangel Henriques
Univ. de Minho
Braga
prh@di.uminho.pt

Maria Joao Varanda
Inst. Politécnico de Bragança
Bragança
mjoao@ipb.pt

Abstract

In this communication in the form of poster and demo we intent to present a tool aimed at evaluating and understanding routing algorithms. This application has a powerful environment that allows: to draw graphs and manipulate their components in manual or automatic form; to incorporate routing algorithms; to make experiments, measurement, and performance tests; to understand programs. The tool incorporates mechanisms to make easy the task for building executable files. These characteristics make this tool a versatile and easy evaluation and comprehension system to be used with routing algorithms.

Along the paper we will emphasize how we conceived and implemented the system's interface to satisfy the user's requirements. Our purpose in the context of Interacçao06 is to demonstrate it!.

Key words

Routing Algorithms, Graph Class, Metrics, Visualization, Comprehension Functions

1. INTRODUCTION

A graph is a tuple $G=(V,E)$ where V is a set of vertices and E is a binary relation defined on V. A graph is a very useful mathematical model to represent real systems (like networks, dependencies, etc.). There are many well known and optimized algorithms to implement the most usual operations over graphs. A routing algorithm is a program used to find one or more paths between two nodes of a graph.

Nowadays there are many routing algorithms tuned to solve each different case effeciently. For example, the Dijkstra's algorithm solves the problem of finding the shortest path between two nodes of a graph. However, the technological advances create new application contexts where traditional algorithms can't be applied; for example the mobile networks. In these applications the network's nodes change of place continuously and therefore the graph that represents the network is unpredictable, it changes dynamically. To solve this problem the scientific community created new routing strategies; for instance the online routing algorithms. The problem is that we don't know the performance of these new routing procedures and sometimes their operation and implementation is not clear.

To attack this problem we conceived a system to experiment, evaluate and understand routing algorithms. The evaluation is carried out through the execution of experiments programmed by the user. To aid the user in understanding the algorithms, the system provides several views of the program under analysis, from the control flow to the machine code.

This communication is organized as follows. Section 2 describes the functional requirements of the routing system. Section 3 shows the approach used to visualize graphs and paths. Section 4 explains the mechanism of automatic compilation and the interaction with other tools. Finally, section 5 presents the comprehension facilities.

2. REQUIREMENTS

In this section we describe the requirements expected from a good system for evaluation/comprehension of routing algorithms. The selection of the criteria is based on the study of the state of the art and on the experience of our research group.

A routing evaluation/comprehension system must provide the following main functions:

1. Visualization of the graph and the path found by each routing algorithm.
2. Incorporation of different routing algorithms, support for graph classes, metrics and experiments.
3. Embedded compilation, linkage and loading processes.
4. Interaction with other tools, such as compilers, spread sheets, text editors, etc.
5. Algorithm Comprehension processes.

For the implementation of these requirements, we built a graphical environment composed by a menu with tool bars and windows that integrate all theses function.

In the following sections we justify theses needs and we describe our solution.

3. VISUALIZATION OF GRAPHS AND PATHS

The visualization of the different kind of graphs and paths is necessary because the user needs to analyze the network type used and to study the routing strategies.

To satisfy this requirement, the evaluator has a window that shows the graph used before having applied the routing strategies (see figure 1).

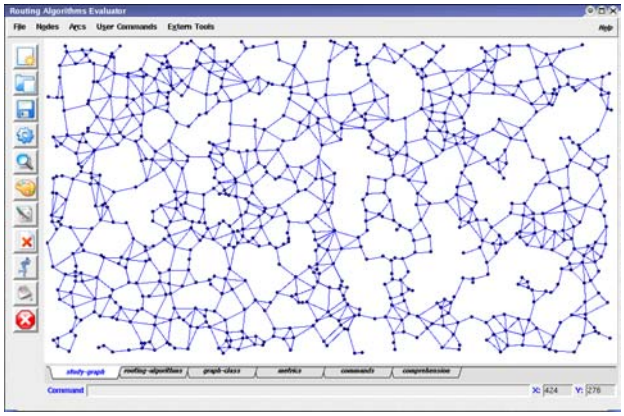

Figure 1: Graph visualiztion

Additionally, the user can see the path found by the routing algorithm chosen. The result can be visualized in two forms: i) as a trace; or ii) as a whole. The first one allows the user to decide step by step when to show the next node in the path. This task is made by using a dialog box (see figure 2).

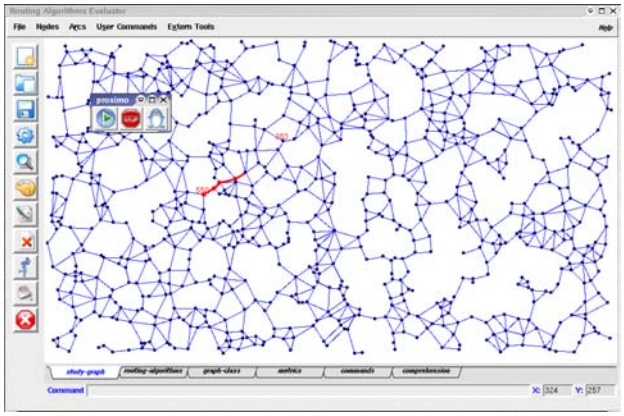

Figure 2: Path visualization using trace method

With the second approach it is possible to see the path without intervention of the user (see figure 3). Furthermore, the evaluator provides functions to: delete; personalize the color; identify and search nodes and arcs. These tasks are made in a simple form using the menu and tool bars.

Inside the graph visualization window, the user can create a graph manually, just clicking with the mouse on the drawing area. When the user wants to create arcs, he must make a click on the determinated area for the source node. Then, without liberating the button, the mouse's pointer must be placed on the determinated area, for target node. Finally the mouse's button can be released.

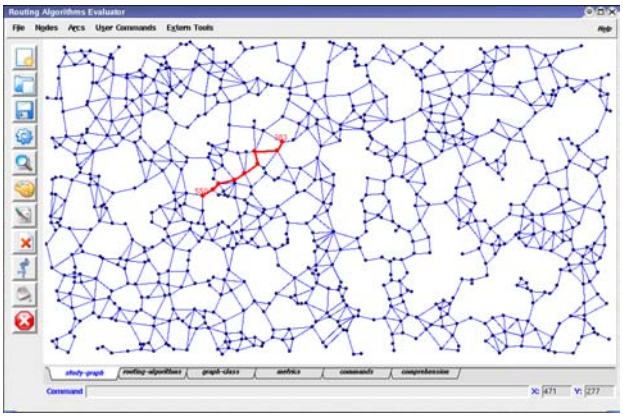

Figure 3: Walk visualization total

3.1 Incorporation of Routing Algorithms, Graph Classes, Metrics and Experiments

In the set of mandatory characteristics that an evaluation/comprehension system must have for routing algorithms, we include operations to incorporate the routing algorithms, graph classes, metrics and experiments easily. In this way, the user is able to focus on the study of routing algorithm study.

To satisfy these requisites, the routing system has four windows to make available each one of these components. Each window has a set of parameter that have to be defined.

For the routing algorithm case the parameters are: routing algorithm's name; function's name which implements it; and the function parameters. After having satisfied the parameter values, the user, presses the *apply* button and the system incorporates this routine to its kernel. Finally, he must press the *build object* button to build the object module of each routing algorithm.

Note that the routing algorithms can be eliminated using the button *delete*. Figure 4 shows the window for the routing algorithm incorporation.

Figure 4: Routing Algorithm window

The other three windows have a similar behavior. The experiment's window has an additional function that allows the automatic creation of dialog boxes. This characteristic has the objective of facilitating the tool usage by the experts. The commands can be executed from the command line or the dialog boxes.

4. AUTOMATIC COMPILATION AND INTERACTION WITH OTHER TOOLS

One of the main difficulties found in systems like this, is the complexity of the routines compilation process because the system has many routines in its kernel. Theses routines implement different data structures useful for the creation of new routing algorithms, graph classes, etc. In addition to this, the possibility that the user has to increment the sysem functions (remember that the user can incorporate new routing algorithms, graphs, metrics, etc.) produces an incresase in the system´s routines. For this reason, the system has a compilation mechanism based in the creation of object modules of each routine that then they are linked to build the executable file. All these tasks are easily made by pressing the button *build object* in the case of routing algorithms, graph classes and metrics. To experiment, the user just has to press the *executable* button. In this way, he is only concerned with the data parameters, and the system makes the rest.

4.1 Interaction with other tools

When the user studies the routing algorithms, he needs: i) to program the routing strategies and ii) to visualize the result. To make these tasks easier the system interacts with other tools such as: compilers, text editors, spread sheets. To access to each tool, the user only needs to activate the option *external tools* of the menu bar and then select the adequate application.

5. COMPREHENSION OF ROUTING ALGORITHMS

Sometimes the programmer has access to routines which implement a task like graph classes, routing algorithms, metrics or experiments that he needs to include in a certain application. However, usually he prefers to develop the program from scratch instead of reusing the one available because reusing implies to understand the code written by another person and that is usually a very hard task. To help in this task, our system has a set of functions aimed at program understanding.

The program comprehension can be reached building different views of the algorithms under study.

We thought that the following views are important: i) the result, that is to say, the output provided by the program; ii) the functions used to get the result and iii) the source and object code associated to each function.

The result is important because it allows the user to have an idea of the algorithm's objective. The list of function calls offer information about how the result was obtained. The two remaining views let that the function's implementation to be visualized and studied. The reader can see the comprehension screen with its four windows in figure 5.

In addition, the system provides navigation functions. Once obtained the list of function calls used to build the result, the user can observe its source and object code just clicking in the name of each function; the system reacts showing the code of the selected function automatically.

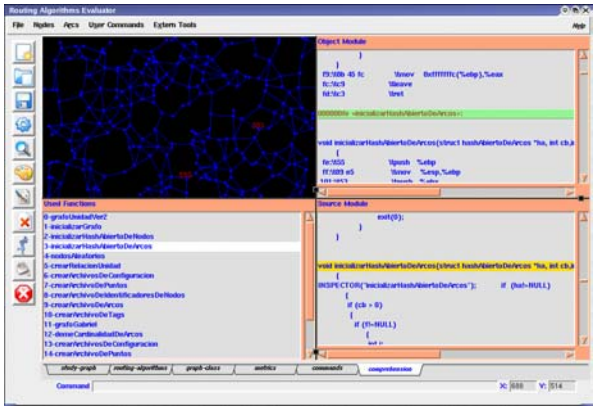

Figure 5: Comprehension Windows

On the other hand, the system permits the user to document each code (source and object) doing click on the corresponding line.

6. REFERENCES

[Berón05] Berón, M; Hernádez Peñalver, G; Gagliardi, O. "Un Evaluador de Algoritmos de Ruteo". Master Thesis in Software Engineering. Universidad Nacional de San Luís. 2005

[Brooks78] Brooks, R. "Using Behavioral Theory of Program Comprehension in Software Engineering". IEEE. 1978.

[Riesco02] Riesco, D; Grosso, A; Berón, M. "Una Herramienta para la Ingeniería Inversa de Sistemas Escritos en Lenguaje C, Bajo Linux". Comunicación, Expuesta y Publicada en el Acta de Resúmenes del Congreso Argentino de Ciencias de la Computación CACIC en el año 2002.

[Linter03] Linter, R; Michand, J; Storey, M; Wu, X. "Plugging-in Visualization: Experiences Integrating a Visualization Tool with Eclipse". ACM Symposium on Software Visualizaton. 2003.

[Mayrhauser95] Mayrhauser, A; Vans, M. "Program Comprensión During Software Maintenace and Evolution". IEEE. 1995.

[Moreno04] Moreno, A; Myller, N; Satinen, E; Ben-Ari, M. "Visualizing Programs with Jeliot 3". ACM. 2004.

[Oliveira06] Oliveira, E; Henriques, P; Varanda, M. "Características de um Sistema de Visualização para Compreensão de Aplicações Web através de Inspecção de Software e baseadas em Modelos Cognitivos". Tesis de Maestría en Ingeniería del Software. 23 de Febrero de 2006.

[Pacione03] Pacione, M; Roper, M; Wood, M. "A Comparative Evaluation of Dynamic Visualization Tools". Proceedings of the 10th Working Conference on Reverse Engineering (WCRE'03). IEEE. 2003.