EULERPATHSOLVER: A NEW APPLICATION FOR FLEURY'S ALGORITHM SIMULATION

Gloria Sánchez–Torrubia, Carmen Torres–Blanc, Leila Navascués-Galante

Abstract: EulerPathSolver is a new application, that meets eMathTeacher specifications and simulates Fleury's algorithm execution. The application runs in a Java Web Start Window and features an animation of the algorithm code, a framework working panel showing the algorithm structures and allowing their manipulation, Pop-up questions, language selector and, save/load options together with the interactive simulation within automatic correction of the user's inputs. It has been designed with the main purpose of supporting active learning as well as being a good aid for teachers when explaining the algorithm process. EulerPathSolver enhances dramatically the old Fleury's Algorithm tutorial designed by the authors and will be a great partner when learning Fleury's Algorithm.

Keywords: eMathTeacher, eLearning, active learning, interactive Java applications, discrete mathematics learning, algorithm visualization.

ACM Classification Keywords: K.3.1 [Computers and Education]: Computer Uses in Education – computer-assisted instruction (CAI), distance learning. K.3.2 [Computers and Education]: Computer and Information Science Education – computer science education, self-assessment. G.2.2 [Discrete Mathematics]: Graph Theory – graph algorithms, path and circuit problems.

Introduction

Since about 13 years ago we have been teaching a Discrete Mathematics course (for first semester undergraduate CS students) that includes an extensive graphs chapter. In the meanwhile we have been changing our goals and methodology, turning this subject to mainly algorithm oriented. The background the students bring, both from their previous learning processes and from their social environment, has transformed their reasoning abilities. Our teaching experience shows that Engineering students have significantly increased their algorithmic reasoning capability, both in comprehension and design, tallying with an important decrease of their formal and algebraic reasoning ability. Thus, fostering graphs' algorithmic approach should be decisive on enhancing students' logical potentials.

Moreover, visualization technologies have been proved as a very positive aid to the learning task, when designed and used under the appropriate conditions [Naps 2003a]. However, comprehensive research is required to determine the best methodology to be applied to the design and development of computer-assisted training, as well as the efficiency of the teaching/learning processes based on this particular method of instruction [Hundhausen 2002].

We began by designing a step by step animated visualization tool (simulating Dijkstra's algorithm) [Sánchez-Torrubia 2001], but immediately discovered that our first semester students adopted a passive attitude that was not beneficial at all for their training. Thus we wondered what those tools should accomplish to avoid this passive attitude and concluded that they should act as "nagging" teachers working next to the student. In our opinion, a teacher should encourage learners to get the concept before starting to practice, and, while practicing, should do nothing but expecting the students' response and guide them towards the right solution. This has been our model when defining what a good tool for actively learning mathematics should accomplish. Therefore, our design philosophy, described by eMathTeacher definition and requirements, has always been focused on the student's interactive prediction, i.e. on the algorithm simulation, helped by the visualization.

eMathTeacher concept

An eLearning tool is eMathTeacher compliant if it works as a virtual maths trainer [Sánchez-Torrubia 2007a]. In other words: it has to be an on–line self–assessment tool that helps students to actively learn mathematic concepts or algorithms by themselves, correcting their mistakes and providing them with clues to find the right solution. The most important feature of these tools is the feasibility to be used in order to practice while the system guides the user towards the right answer. A tool, designed under this philosophy, should also be useful as bLearning (blended learning) complementary material both for being used by teachers in classroom lectures and by students when learning math by themselves.

The aforementioned idea needs to be concreted by means of a list of features that can be implemented. We list the main requirements we have developed to define what a tool should accomplish for attaining this goal.

- *Step by step inquiring*: for every process step, the student should provide the result of the current step execution whilst the application waits in a stand by mode, expecting the user's input.
- *Step by step evaluation*: just after the user's entry, the tool evaluates it, providing a tip in case it is wrong, or executing it when it is ok.
- *Visualization of the step's changes*.
- S*elf–assessment levels* option, implementing correction after several steps or even after several iterations.
- *Pop–up questions* assessing the comprehension of the underlying algorithm principles, thus enhancing in–deep understanding and not only process learning.
- *Animated Algorithm code visualization panel*.
- *Framework* working *panel* showing the current state of the algorithm data structures. It should also allow the user to update those structures.
- *Saving/retrieving* option, data library and/or automatic exercises generation.
- *Language menu*.
- *Easy to use. Clear presentation within a nice and friendly graphic environment*.
- *Flexible and reliable*: allowing users to introduce and modify the example and to repeat the process if desired.
- *Platform independency and continuous availability* (anytime, anywhere).
- *No installation or maintenance* tasks required and low downloading size.

As members of GIDA^{2}M, -an education research group- our work is mainly oriented to design and develop applications for actively learning mathematics following the design methodology that we have called eMathTeacher. We have developed several applications, designed under this philosophy, for actively learning graph algorithms. The applications correspond to DFS & BFS, Fleury's, Prim & Kruskal's and Dijkstra's (PathFinder) algorithms [Sánchez-Torrubia 2007a & Sánchez-Torrubia 2009]. Additionally, Mamdani's eMathTeacher [Sánchez-Torrubia 2008b] is a tutorial for actively learning this fuzzy inference method and it has also been designed under the eMathTeacher philosophy. The tools, implemented as Java applications, are available in GIDA^{2}M website http://www.dma.fi.upm.es/gies/gidam/home.html#aplicaciones

As mentioned above, we have designed several tools, mainly implementing graph algorithms, and have measured their impact on students' learning [Sánchez-Torrubia 2007b]. It has been evaluated by comparing the marks obtained by two different groups of students in the graph exercise of a Discrete Mathematics final exam. In this exercise, the students had to apply two algorithms on a particular graph. The study group used the applications for bLearning while the control group did not use them. Data showed a deeper understanding of algorithms process in the study group, with the highest marks percentage clearly higher.

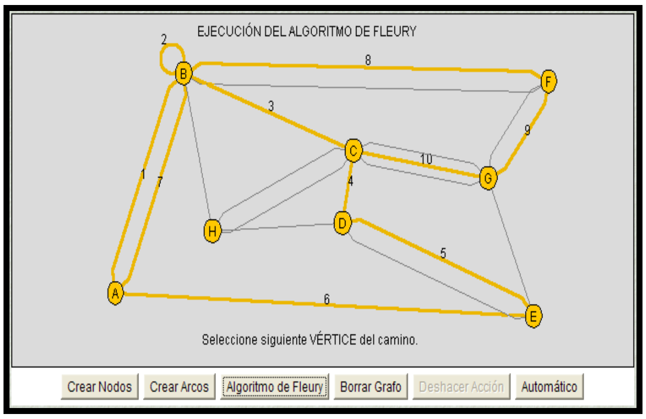
Fleury's algorithm and first interactive tool

Fleury's algorithm is designed for finding an Euler Path in an undirected graph. The graph has an Euler path if it is possible to start at a vertex and move along the graph so as to pass along each edge without going over any of them more than once.

In [Euler 1736], Euler states the following result: A finite graph G contains an Euler path if and only if G is connected and contains at most two vertices of odd degree. He also sketches a procedure for finding the path consisting on creating a simple circuit, eliminating the used edges, finding new circuits in the remaining graph and joining the new circuits in the proper vertexes. This procedure, very intuitive in theorem's formal demonstration, is not algorithmically effective.

The algorithmic solution to the problem of finding an Euler Path is credited to a French mathematician named Fleury [Fleury 1883]. The basic idea is that when drawing an Euler circuit, all passed edges cannot be used again. So, at any moment in drawing, with all passed edges deleted, the remaining edges must be in one connected component. It starts with a vertex of odd degree —if the graph has none, then start with any vertex—. At each step it moves across an edge which is included in a cycle, unless there is no choice, and then we delete that edge.

The fist tool designed under eMathTeacher philosophy simulating Fleury's algorithm (see Figure 1) consisted of an interactive Java applet that implemented it and has been in service for several years. Within this applet the user introduced the graph and simulated the algorithm execution while the application evaluated the provided inputs. In other words: in real time, the applet only evaluated the input introduced by the user. If it was right, the application implemented the order, and then it remained in a stand by mode, waiting for a new one. If the input was not right, an error message appeared on the message window, indicating to the user, what the error was and waiting for the right one. Once the algorithm has been completed, a successful '*end of algorithm*' message was displayed [Sánchez-Torrubia 2008a].


EJECUCIÓN DEL ALGORITMO DE FLEURY

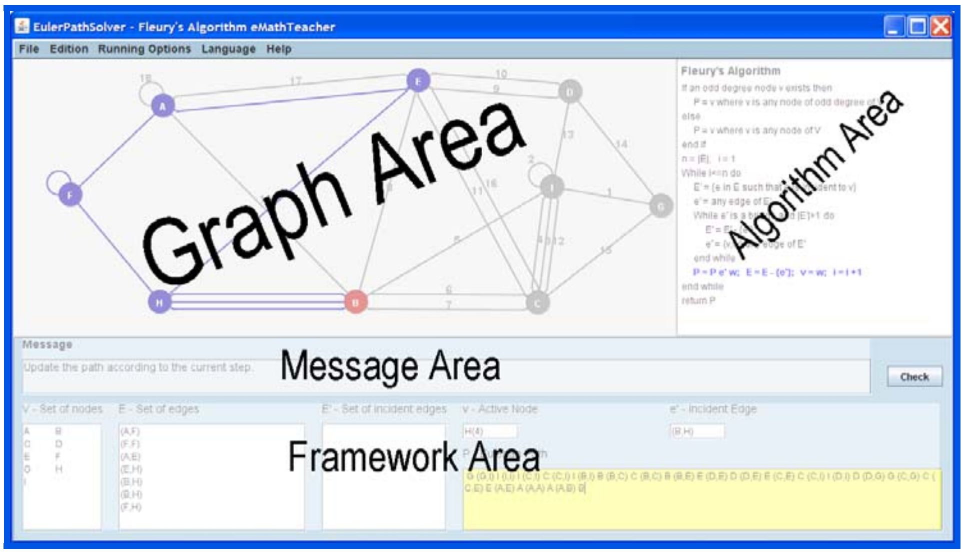
Seleccione siguiente VÉRTICE del camino.

Figure 1: Old Fleury's applet in the process

New EulerPathSolver application

The applet described in the previous section is a tool that was implemented some years ago and, in the meanwhile, there are some features we lacked. Thus, we have designed a new Java application solving those problems and adding new features:

- One of the Java applets' deficiencies is that, due to security reasons, they are not allowed to write in the client's hard disk. It means that users cannot save their work for later review and every time they want to practice, they have to create the graph. The problem has been solved by using Java Web Start.
- Another feature we missed is related to language capabilities. The first applet was implemented only in Spanish. The new application includes a language menu, starting with Spanish and English and expecting to get some support for being translated to other languages.
- CS students should get used to reading, understanding and executing algorithm code instructions. To achieve this goal with the first tool we encouraged them to read the algorithm code while practicing the algorithm. The Animated Algorithm Code Visualization panel provides a better understanding of the algorithm execution code as it shows the current instruction by changing its colour. Additionally, when the user makes a mistake, the corresponding instruction is shown in red.
- The old tool did not show the algorithm structures at all which made the comprehension of the internal operations of the algorithm difficult. The inclusion of a framework working panel allows users to practice each algorithm step exactly as if they were simulating it by hand, working with the algorithm structures.
- Some of the working details of the algorithm, or the reasons why it works, may remain hidden in the mechanic execution of the pseudo code. The implementation of Pop-up questions, emphasizing these details. will improve the deep understanding of the algorithm.



```
Fleury's Algorithm
If an odd degree node v exists then
    P = v where v is any node of odd degree of V
else
    P = v where v is any node of V
end if
n = |E|;  i = 1
While i<=n do
    E' = {e in E such that [illegible] incident to v}
    e' = any edge of E'
    While e' is a b[illegible] and |E|+1 do
        E' = E' - {e[illegible]
        e' = (v, [illegible] edge of E'
    end while
    P = P e' w;  E = E - {e'};  v = w;  i = i + 1
end while
return P
```

Message

Update the path according to the current step.

V - Set of nodes: A B C D E F G H I

E - Set of edges: (A,F) (F,F) (A,E) (E,H) (B,H) (B,H) (F,H)

E' - Set of incident edges

v - Active Node: H(4)

e' - Incident Edge: (B,H)

G (G,I) I (I,I) I (C,I) C (C,I) I (B,I) B (B,C) C (B,C) B (B,E) E (D,E) D (D,E) E (C,E) C (C,I) I (D,I) D (D,G) G (C,G) C (C,E) E (A,E) A (A,A) A (A,B) B

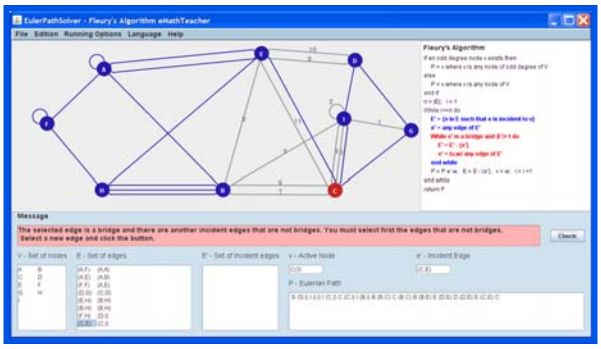
Figure 2: EulerPathSolver new panels

Tool description

EulerPathSolver is launched by a *jnlp* file and opens in a Java Web Start window.

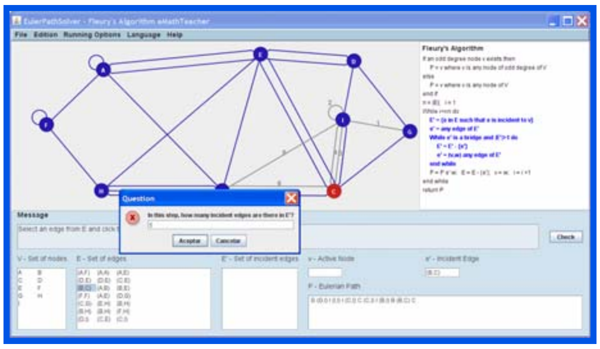
The window is split into four areas: graph, algorithm, messages and framework areas (see Figure 2).

- The *graph area* displays the graph and allows graph edition. When the algorithm is being simulated, the colours of nodes and edges change according to the algorithm execution.
- The *algorithm area* displays the execution code, showing in blue the current step or in red the point where the user's mistake is located,
- The *message panel* provides clues to find the right solution or indicates the next step to be done. This panel also offers useful hints when the graph is being edited.
- The *framework* working *panel* shows the structures current state and allows interaction to simulate the algorithm execution.


EulerPathSolver - Fleury's Algorithm eMathTeacher
File Edition Running Options Language Help
Message
Check
V - Set of nodes
E - Set of edges
E' - Set of incident edges
v - Active Node
e' - Incident Edge
P - Eulerian Path

The selected edge is a bridge and there are another incident edges that are not bridges. You must select first the edges that are not bridges.
Select a new edge and click the button.

Figure 3: EulerPathSolver showing an error.

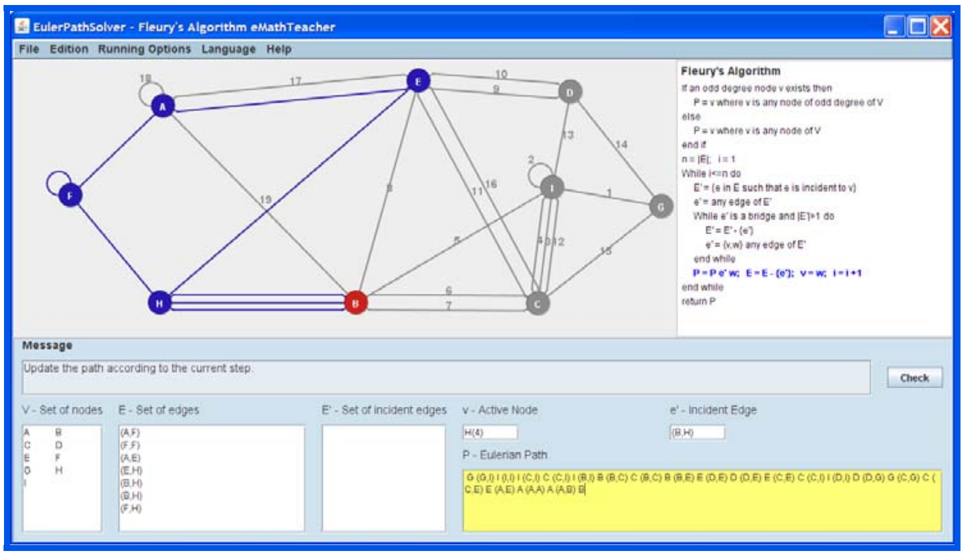

EulerPathSolver - Fleury's Algorithm eMathTeacher
File Edition Running Options Language Help
Question
Aceptar
Cancelar
Message
Check

In this step, how many incident edges are there in E'?

Figure 4: Pop-up question.

The menu bar features a file menu including graph saving/loading and library saving/loading options, edition menu containing create/delete node, create/delete edges, delete graph, undo and redo, two execution modes and a process interruption option. There is also a language selector, currently implemented in Spanish and English.

Once the graph has been introduced and the algorithm is running, the application checks whether an Eulerian path exists. If the graph is not connected or there are more than two odd degree vertexes, an error message is displayed indicating the corresponding fault. If there is an Eulerian path the simulation starts. In each step of the process, the tool shows the current state of the algorithm structures: nodes, edges, active node, edges which are incident with the active node (only in the demonstrative mode) and the covered path.


EulerPathSolver - Fleury's Algorithm eMathTeacher
File Edition Running Options Language Help
A
B
C
D
E
F
G
H
I
Check

Fleury's Algorithm

```
If an odd degree node v exists then
   P = v where v is any node of odd degree of V
else
   P = v where v is any node of V
end if
n = |E|;  i = 1
While i<=n do
   E' = {e in E such that e is incident to v}
   e' = any edge of E'
   While e' is a bridge and |E'|>1 do
      E' = E' - {e'}
      e' = (v,w) any edge of E'
   end while
   P = P e' w;  E = E - {e'};  v = w;  i = i+1
end while
return P
```

Message

Update the path according to the current step.

V - Set of nodes: A B C D E F G H I

E - Set of edges: (A,F) (F,F) (A,E) (E,H) (B,H) (B,H) (F,H)

E' - Set of incident edges:

v - Active Node: H(4)

e' - Incident Edge: (B,H)

P - Eulerian Path: G (G,I) I (I,I) I (C,I) C (C,I) I (B,I) B (B,C) C (B,C) B (B,E) E (D,E) D (D,E) E (C,E) C (C,I) I (D,I) D (D,G) G (C,G) C (C,E) E (A,E) A (A,A) A (A,B) B

Figure 5: EulerPathSolver updating the Eulerian path.

In the interactive mode, the user should select the next edge to be included in the Eulerian path and, sometimes, add the edge and the new active node to the covered path manually (see Figure 5). For every step, when the user makes a mistake, an error message appears in the message area while the correspondent line within the algorithm code is highlighted in red (see Figure 3). Randomly Pop-up windows appear asking questions related to the current step (see Figure 4).

When the iteration has been completed the application changes the selected edge's color and adds a number indicating the path sequence (see Figure 5).

Conclusion

In the field of graph algorithms we are designing new tools, as well as updating and enhancing the oldest ones. EulerPathSolver meets the specifications listed in the introduction, which means it is eMathTeacher compliant. Actually, the highlighting new feature of this application is an animated algorithm visualization panel, able to display, on the code, the current step and the user's mistake. Other relevant new attribute included is an active framework area for the algorithm data, where the user can modify the algorithm structures while the application verifies the correctness of the input. Finally, it offers the feasibility of saving and retrieving graphs and the graph file follows XML standards.

EulerPathSolver has been designed with the main purpose of supporting active learning (in interactive mode) as well as being a good aid for the teacher when explaining the algorithm process (in demonstrative mode). The new application enhances dramatically the old Fleury's Algorithm tutorial [Sánchez-Torrubia 2008a] and will be a great partner when learning Fleury's Algorithm.

Bibliography

[Euler 1736] Euler, L. Solutio problematis ad geometriam situs pertinentis. Commentarii Academiae Scientiarum Imperialis Petropolitanae 8, 128-140 (1736). Based on a talk presented to the Academy on 26 August 1735.

[Fleury 1883] Fleury. Deux problemes de geometrie de situation. Journal de mathematiques elementaires 1883, 257-261.

[Hundhausen 2002] Hundhausen, C. D., Douglas, S. A. and Stasko, J. T. A Meta-Study of Algorithm Visualization Effectiveness. Journal of Visual Languages and Computing, 13, 3, Elsevier, 259-290 2002.

[Naps 2003a] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., Velazquez-Iturbide, J. A.: Exploring the Role of Visualization and Engagement in Computer Science Education. Inroads - Paving the Way Towards Excellence in Computing Education. pp. 131-152, ACM Press, 2003.

[Sánchez-Torrubia 2009] M.G. Sánchez–Torrubia, C. Torres–Blanc and M.A. López–Martínez, PathFinder: A Visualization eMathTeacher for Actively Learning Dijkstra's algorithm. Electronic Notes in Theoretical Computer Science, 224 (1), Elsevier, 151-158 (2009).

[Sánchez-Torrubia 2008a] M.G. S□anchez–Torrubia, C. Torres–Blanc, and V. Giménez–Martínez. An eMath-Teacher tool for active learning Fleury's algorithm. International Journal Information Technologies and Knowledge (IJ ITK), 2(5):437-442 (2008).

[Sánchez-Torrubia 2008b] M.G. Sánchez–Torrubia, C. Torres–Blanc and S. Krishnankutty, Mamdani's Fuzzy Inference eMathTeacher: a tutorial for Active Learning. WSEAS Transactions on Computers 7 (5), 363–374 (2008).

[Sánchez-Torrubia 2007a] M. G. Sánchez–Torrubia, C. Torres–Blanc, J. Castellanos, New Interactive Tools for Graph Algorithms Active Learning. ACM SIGCSE Bulletin, 39 (3), 337 (2007).

[Sánchez-Torrubia 2007b] M. G .Sánchez–Torrubia, C. Torres–Blanc, J. Castellanos, Defining eMathTeacher Tools and Comparing them with e&bLearning web based tools. Proceedings of the 2007 International Conference on Engineering and Mathematics (ENMA 2007). (Bilbao, Spain, 7-9 July 2007).

[Sánchez-Torrubia 2001] M.G. Sánchez–Torrubia, V. Lozano–Terrazas, Algoritmo de Dijkstra: Un tutorial interactivo. Proceedings of the VII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2001). (Palma de Mallorca, Spain, 16–18 July, 2001). J. Miró, 254-258.

Authors' Information

Gloria Sánchez–Torrubia – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: gsanchez@fi.upm.es

Carmen Torres–Blanc – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: ctorres@fi.upm.es

Leila Navascués-Galante – Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s.n., 28660 Boadilla del Monte, Madrid, Spain; e-mail: leila.navascues@gmail.com