ANNALS of Faculty Engineering Hunedoara
International Journal of Engineering
ISSN: 1584 - 2673

EDUCATIONAL SOFTWARE METHODS AND STRATEGIES FOR DESIGNING ALGORITHMS BACKTRACKING, GREEDY METHOD AND DYNAMIC PROGRAMMING

Tatiana-Elena DUȚĂ, Alexandru OPREAN, Manuela PĂNOIU

Department of Electrical Engineering and Industrial Informatics,
"Politechnica" University of Timisoara, ROMANIA

Abstract

This paper present visual interactive software which shows through simulation the main algorithms use in searching solutions in artificial intelligence. It was studies backtracking, greedy and dynamical programming algorithms. The software is multimedia educational software designed for acquisition and learning process and it can be seen as a modern method for teaching methodology. It was also perform a comparative study of these algorithms using some typical known problems.

Keywords:

Algorithm visualization, backtracking, dynamic programming, greedy algorithm

1. INTRODUCTION

The teaching art consists not only in the scientific strictness of exposing the subject matter to be studied, but also in including in this activity of all influence forms upon the student [1]. One of the main aspects of using computer for lessons is the development of the students creative thinking. An optimal mean in this case is the introduction in the computational training means of the interactivity elements. Educational software provides this important feature - it is interactive: it offers to the learner the opportunity to manipulate the model for achieving in a short time, a high volume of knowledge (more complex and stable).

This paper present not only one useful application to the study of solving algorithms with backtracking, greedy method and dynamic programming but also a comparative study of these algorithms in terms of executive time.

2. DESCRIPTION OF THE INFORMATICS SYSTEM

For informatics system design it was use a visual oriented object language, Borland C++ Builder. This environment is very useful because it generate a native code for Windows platform, which is the most used operating system.

Application interface is very simple. I have focused on the fact that all students and even beginners, in algorithms and programming tricks could use it. From the main menu, you can see in Figure 1, you can choose four paths of the project:

- Presentation of Backtracking method
- Presentation of Greedy method
- Presentation of Dynamic Programming
- A comparative study for these three algorithms earlier selected

By selecting any of these options from the application interface a new window will open which will contain the main menu for accessed algorithm.

The animation is realized using Xara 3D and starts the creation of the form.

Buttons are simulations realized with the help of images performed through program 3D Button. The effect of "push" the buttons is realized with Label type components, showing the explanatory text of the buttons.

Here are the main buttons for navigation. The name of each button will show the exact segment you want to make active. So, every time you click on *, from the book, you are accessing the" Theory"

page where you'll find three more buttons. Each button is a links for another page, where you can find, depending on the button name, information about the accessed method.

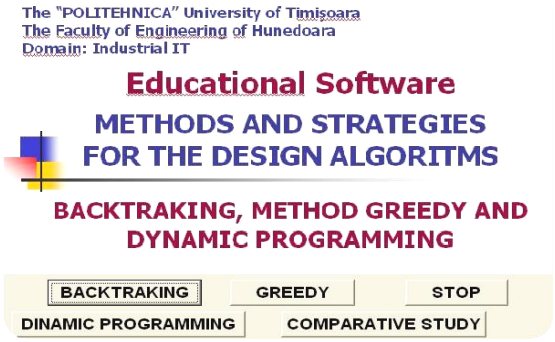

Figure 1. Main Menu

By running a click on Applications, you'll find applications, made to exemplify through images and algorithm of this procedure. Also, through the window on the right corner of the interface, you can learn what each button contains, simply passing the mouse over it.

2.1.Backtracking algorithm simulation

Backtracking is a general algorithm for finding all (or some) solutions to some computational problem, that incrementally builds candidates to the solutions, and abandons each partial candidate c ("backtracks") as soon as it determines that c cannot possibly be completed to a valid solution, [3].

Backtracking is an important tool for solving constraint satisfaction problems. It is often the most convenient (if not the most efficient) technique for parsing, for the knapsack problem and other combinatorial optimization problems. It is also the basis of the so-called logic programming languages such as Prolog use for artificial intelligence. Backtracking can be applied only for problems which admit the concept of a "partial candidate solution" and a relatively quick test of whether it can possibly be completed to a valid solution. When it is applicable, backtracking is often much faster than brute force enumeration of all complete candidates, since it can eliminate a large number of candidates with a single test.

For backtracking algorithm simulation it was use the following well known problems: the N queen's problem, the coloring maps problem, the horse jump on chessboard problem, and other problem that can be solve using backtracking. From these one, in this paper is present the map coloring problem.

The map coloring problem is a very well known problem and consists in finding the appropriate color to coloring a map so than two neighborhood regions on the map to have distinct colors. The graphical user interface for algorithm analyze in shown in fig. 3.

Figure 2. Main Menu Backtracking

The simulation allows to visualize the algorithm step by step or to visualize entire algorithm. For the first case the user can push the "Step by step" button after each step. This means that appear a new color for the current region on the map, after pushing the above mentioned button. At the same time the stack variation is updated by increasing or decreasing the top of stack. During the simulation the user can see the stack variation and the map partially colored. The test map can be load from a file (a *.bmp file). In a frame is show also the algorithm implemented in C++. After a solution is found, this is displayed in a separate frame.

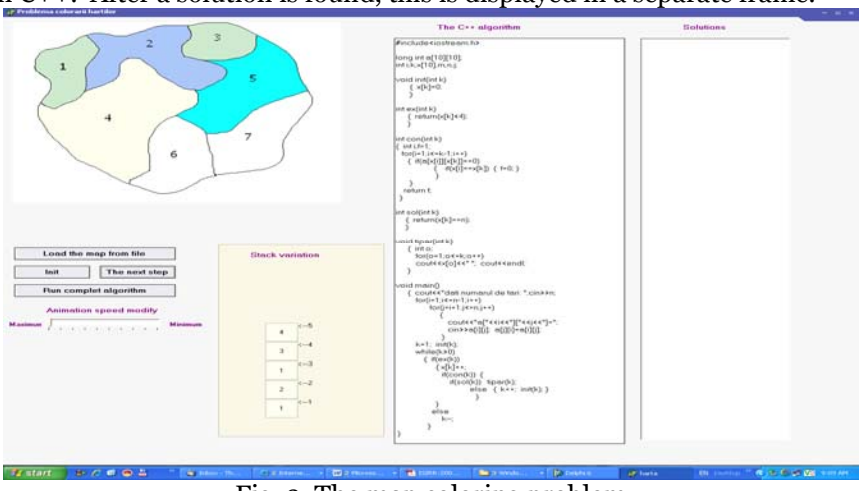

Fig. 3. The map coloring problem

2.2. Greedy algorithm simulation

A greedy algorithm is any algorithm that follows the problem solving meta-heuristic of making the locally optimal choice at each stage [4] with the hope of finding the global optimum. For example, applying the greedy strategy to the traveling salesman problem yields the following algorithm: "At each stage visit the unvisited city nearest to the current city". Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. Nevertheless, they are useful because they are quick to think up and often give good approximations to the optimum.

For Greedy algorithm simulation it was use the following well known problems: Dijkstra's Algorithm, Connecting cities with minimum cost, Prim's Algorithm, the Knapsack problem. These examples can be selected using a graphical user interface similar with the one presented in fig. 2.

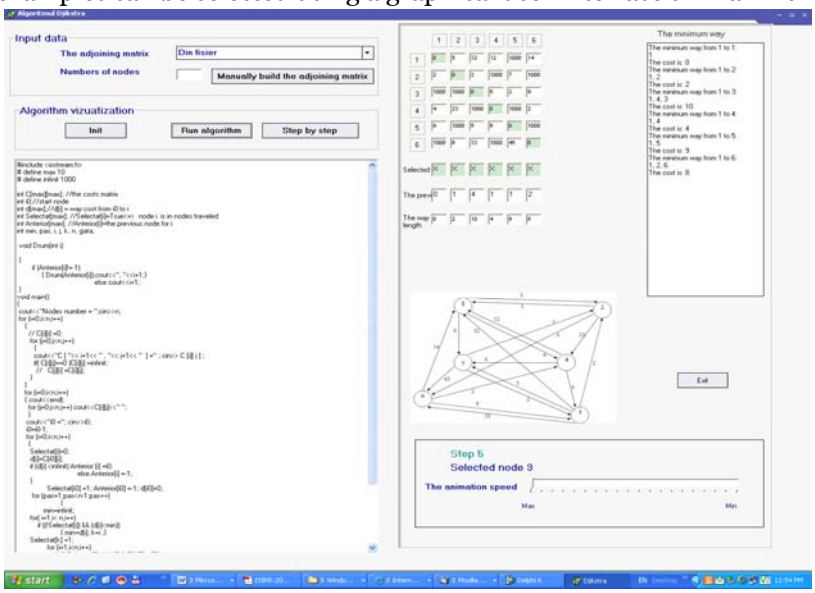

Fig. 4. The Dikstra's algorithm visualization

The Dikstra's algorithm visualization is exemplified in this paper. The graphical user interface is shown in fig 4.

In this simulation, the user can build the adjoining matrix for the graph manually or be loading from a file. As the previous presented simulation, the user can run the entire algorithm using a desired animation speed or the algorithm can be run step by step. During the simulation, the user can see the costs matrix updating, a vector with selected nodes and the current length way in each moment. Finally, in a frame on the GUI, all the ways from all the nodes to the selected start node are printed. The user can see also the algorithm implemented in C++ language.

2.3. Dynamical programming algorithm simulation

In computer science and also in mathematics, dynamic programming is a method of solving problems that exhibit the properties of overlapping subproblems and optimal substructure [5]. The method takes much less time than native methods. This method can be applied to many string algorithms including longest common subsequence, longest increasing subsequence, longest common substring. It can be also applied to many algorithmic problems on graphs can be solved efficiently for graphs of bounded treewidth or bounded clique-width by using dynamic programming on a tree decomposition of the graph. In this software was simulating the Roy – Floyd algorithm. This is a simple algorithm for determining dot-matrix graph roads. One execution of algorithm finds the shortest path between all pair vertices of the graph. In many practical situations the question is to determine a shortest way between two vertices of the graph. The graphical user interface is similar with the GUI for Djiskstra's algorithm simulation and provides the same options. Another simulation was made for maze problem.

3. CONCLUSIONS. COMPARATIVE STUDY

Comparative study was possible after using backtracking algorithms, greedy and dynamic programming of some representatives problems. The conclusions are presented below.

3.1. Backtracking-Dynamic programming

Solving problems by dynamic programming is done in polynomial time, because each optimal "more general" is calculated from optimum "more private" searching in polynomial time, and the calculated optimum time is not recalculated later but switched to calculating the optimum "more general".

Therefore the method of dynamic programming may be considered as an alternative to the backtracking method. It is clear that the problems which may be solved through backtracking may be solved by dynamic programming as well. If the backtracking method is used you can obtain an

algorithm that can reach (in the most unfavorable case) exponentially. Dynamical programming is more efficient that Backtracking.

If the dynamic programming method is used we can obtain a single optimal solution, unlike the backtracking method that generates all the optimal solutions.

3.2. Backtracking-Greedy

Although both methods offer solutions in the form of vector, the greedy method and backtracking method have the following differences:

- backtracking technique provides all the problem solutions, while greedy method is providing a single solution;
- greedy technique doesn't have a mechanism for going back (which is specific for backtracking method) that is why is impossible to achieve the global optimum.

Regarding the time running, Greedy algorithms are more efficient but it doesn't apply to whatever problem.

3.3. Greedy-dynamic programming

Both dynamic programming and greedy technique can be used when the solution to a problem is seen as the result of a sequence of decisions. The essential difference between greedy technique and dynamic programming is that the greedy method generates a single sequence of decisions, exploiting incompletely the optimality principle. In dynamic programming are generated more sub sequential decisions, taking in consideration the optimality principle, but considering only the best sub sequences, combining them in the final optimal solution. Although the total number of sequences of decisions is exponential, dynamic programming algorithms are often polynomial, the reduction of complexity due to the use optimality principle. Another important characteristic of dynamic programming is that it stores the optimum sub sequences, thus avoiding their recalculation.

Although the greedy algorithm does not guarantee obtaining the optimal solution, however it has the advantage that it is more efficient in terms of execution time and memory used than dynamic programming algorithm and the corresponding backtracking method.

Final conclusion is: when solving a problem by greedy method, execution time is polynomial (instant solution) for different input data. Solving through the backtracking method the execution time increases exponentially by the increase of input data's volume. For the dynamic programming method the execution in most cases is increasing polynomial with the input data. If recursively is used in solving a problem using dynamic programming, execution time increases exponentially with the input data.

REFERENCES

[1.] *E. Scalon, C. Tosunoglu, A. Jones, P. Butcher, S. Ross, J. Greenberg, Learning with computers : experiences of evaluation, Computer Education, Elsevier Science, 1998*

[2.] *J. Trindade, C. Fiolhais, L. Almeida, Science Learning in Virtual Environments, British Journal of Educational Technology, 2002*

[3.] *Donald E. Knuth (1968). The Art of Computer Programming. Addison-Wesley.*

[4.] *Introduction to Algorithms (Cormen, Leiserson, and Rivest) 1990, Chapter 16 "Greedy Algorithms" p. 329.*

[5.] *Bertsekas, D. P., 2000. Dynamic Programming and Optimal Control, Vols. 1 & 2, 2nd ed. Athena Scientific.*

[6.] *Herlo D., Metodologia educaţională, Editura Universităţii „Aurel Vlaicu", Arad, 2000*

[7.] *Mihai Oltean, Programare avansată în Delphi, Editura Albastră, Cluj-Napoca, 2000*

[8.] *Pătruţ B., Aplicaţii în Delphi, Editura Teora, Bucureşti, 2001*

[9.] *Sorin T., Iniţiere în programarea vizuală, Editura L&S INFOMAT, Bucureşti, 2002*

[10.] *Vladimir-Ioan Creţu, Structuri de date şi algoritmi, Editura Orizonturi Universitare, Timişoara 2000*