ANNALS OF FACULTY ENGINEERING HUNEDOARA
– International Journal of Engineering
Tome XI (Year 2013) – FASCICULE 3 (ISSN 1584 – 2673)

[1.] Anca-Elena IORDAN

IMPLEMENTATION OF AN INTERACTIVE SOFTWARE DESIGNED FOR THE STUDY OF UNINFORMED SEARCH STRATEGIES

[1.] UNIVERSITY POLITECHNICA OF TIMISOARA, FACULTY OF ENGINEERING HUNEDOARA, ROMANIA

ABSTRACT: In this study the author present the needful stages in object orientated implementation of an interactive software that is dedicated to the process of acquaintances assimilation in artificial intelligence area for the study of uninformed search strategy, particular for simulation and comparing of the following three strategies: breadth first search strategy (BFS), depth limited search strategy (DLS) and depth first iterative deepening search strategy (IDS). The modelling of the environment is achieved through specific UML diagrams representing the stages of analysis, design and implementation. This interactive environment is much helpful for both students and professors, because computer programming area, for the most part artificial intelligence area is comprehended and assimilated with difficulty by students.
KEYWORDS: Uninformed Search Strategies, Breadth First Search, Depth Limited Search, Depth First Iterative Deepening Search, UML, Java

INTRODUCTION

The search aspect is omnipresent in artificial intelligence [1]. The efficiency of a geat many artificial intelligence systems is dominated by the complexity of a search strategy in their inner loops.

Simulation of uninformaed search strategies using classical 8-puzzle game has as goal the accurate comprehending and implementing by students of the next three search strategies: breadth first search strategy (BFS), depth limited search strategy (DLS) and depth first iterative deepening search strategy (IDS) in problem solving, representing a complementary methodological instrument in academical teaching process.

To realize the interactive software designed it was followed the accomplishment of following targets:

- The oriented object design of the software by identifying the concepts who characterize the uninformed search strategies.
- The interactive simulation of the strategies, the emphasis being set on the two sets used by these strategies: open and closed lists.
- Realization of a comparing study of the three strategies from the point of view of the space and time complexity.

UNINFORMED SEARCH STRATEGY - Breadth First Search Strategy

In breadth first search [2] the open list is implemented as a FIFO (first-in, first-out) queue. Hereby, the route which is selected from the open list is the one which was added earliest. This manner implies that the routes from the start node are generated in order of the number of arcs in the route. One of the routes with the fewest arcs is selected at each step.

Breadth-first search is efficacious when:

- space dimension is not an impediment;
- you want to find the solution containing the fewest arcs;
- some solutions may exist, and at least one has a short route length;
- infinite routes may exist, forasmuch it explores all of the search space, although infinite routes.

It is a small strategy when all solutions have a long route length or there is few heuristic knowledge available. It is not used nearly often forasmuch of its space complexity.

Depth Limited Search Strategy

The second strategy is depth limited search. In depth-first search [3], the open list acts like a last-in first-out stack. The states are added to the stack one at a time. The one selected and taken off the open list at any time is the last state which was added.

Depth-first search is appropriate when either space is restricted or many solutions exist, possible by long route lengths, especially for the case where almost all routes lead to a solution. It is a small strategy when it is possible to get pinned in infinite routes. This occurs when the graph is infinite or when there are cycles in the graph.

Analogous with the normal depth first search, depth limited search [4] is an uninformed search. It works ditto with depth first search, but avoids its disadvantages respecting completeness by imposing a maximum limit on the depth of the search. Although the search could still globing a vertex beyond that depth, it will not do so and herewith it will not follow infinitely deep routes. However depth limited search will find a solution if it is within the depth limit, that guarantees at least completeness on all graphs.

Depth limited search is the core for a number of other strategies, such as iterative deepening.

Depth First Iterative Deepening Search Strategy

While still an unintelligent strategy, the depth first iterative deepening search [5] combines the positive concepts of breadth first and depth first searching to create a strategy that is frequent an upgrading over each strategy individually.

A depth first iterative deepening search operates ditto a depth first search, with the exception of skimpy more constrained: there is a maximum depth that defines how many tiers deep the strategy can look for solutions. A node at the maximum tier of depth is treated as terminal, even if it would ordinarily have successor nodes. If a search fails, then the maximum tier is increased by one and the process repeats.

OBJECT ORIENTATED IMPLEMENTATION OF THE SOFTWARE

With the aim to observe the approach of modifying of the two lists that are used for implementing uninformed search strategies: open list and closed list and eke to compare the obtained results, there was implemented an interactive Java application [6]. For object orientated design of the software, unified modeling language will be used. To achieve UML diagrams were used the ArgoUML software [7].

UML Use Case Diagram

The analysis of the informatics system consists in drawing the use case diagrams [8]. The informatics system will be described in a outright and terse approach by representation of the use-cases. Each case describes the interaction among the user and the system. The diagram defines the system's domain, allowing visualization of the proportion and aim of the entire developing action. Use case diagram is represented in figure 1 and includes:

- *One actor - the user who is external entity with that the interactive software interacts.*
- *Six use-cases which describe the performance of the interactive software.*
- *Relationships between user and use-cases (association relationships), and relationships between use-cases (dependency and generalization relationships).*

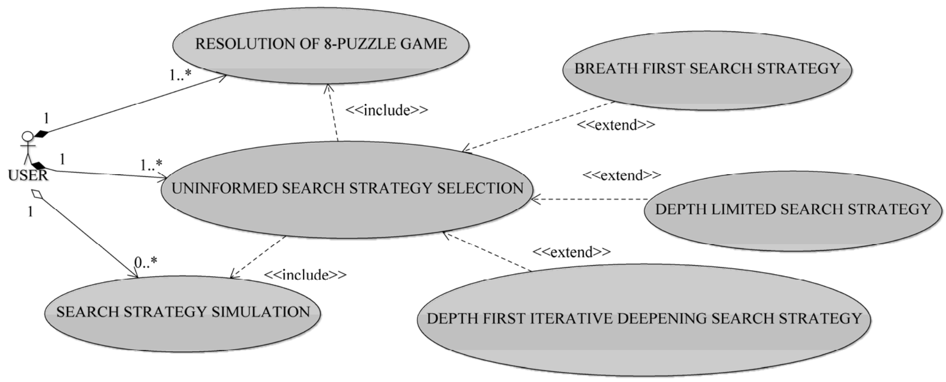

Figure 1. Use Cases Diagram

UML Class Diagram

Conceptual modelling allows identifying the very important elements for the interactive software [9]. Class diagram is represented in figure 2 in order to be observed the connection mode between the classes and the interfaces that are used and also the composition and aggregate relationships between instances.

For memorizing a state configuration, there was implemented "StatePuzzle" class which implements "State" interface. For memorizing a node of search tree, there was implemented "ExploredNodePuzzle" class which realizes "ExploredNode" interface and for memorizing an expanded node together with its successors, there was implemented "ExpandedNodePuzzle" class which realizes "ExpandedNode" interface.

Breadth first search strategy is implemented by using "BFSStrategy" class. An instance of this class is composed by two instances of "ExploredNodePuzzle" class, according to composition *relationship which exists in diagram, but can also contain instances of "ExpandedNodePuzzle" class, as it can be observed from aggregate relationship presented in diagram.*

Depth limited search strategy is implemented by using "DLSStrategy" class. An instance of this class is composed by two instances of "ExploredNodePuzzle" class, according to composition relationship which exists in diagram, but can also contain instances of "ExpandedNodePuzzle" class, as it can be observed from aggregate relationship presented in diagram.

Depth first iterative deepening search strategy is implemented by using "IDSStrategy" class. An instance of this class is composed by two instances of "ExploredNodePuzzle" class, according to composition relationship which exists in diagram, but can also contain instances of "ExpandedNodePuzzle" class, as it can be observed from aggregate relationship presented in diagram.

For realizing the two windows that will compose the graphical interface of the aplication, there were implemented the following two classes: „Resolution" for the main window and „Simulation" for the simulation of the three uninformed search strategy. We can observe that the „Simulation" class implements the „Runnable" interface. An instance of this class is composed by one instance of „Resolution" class, two instances of "StatePuzzle" class and two instances of "ExploredNodePuzzle" class, according to composition relationship which exists in diagram, but can also contain instances of "BFSStrategy", "DLSStrategy" and "IDSStrategy" classes, as it can be observed from aggregate relationship presented in diagram.

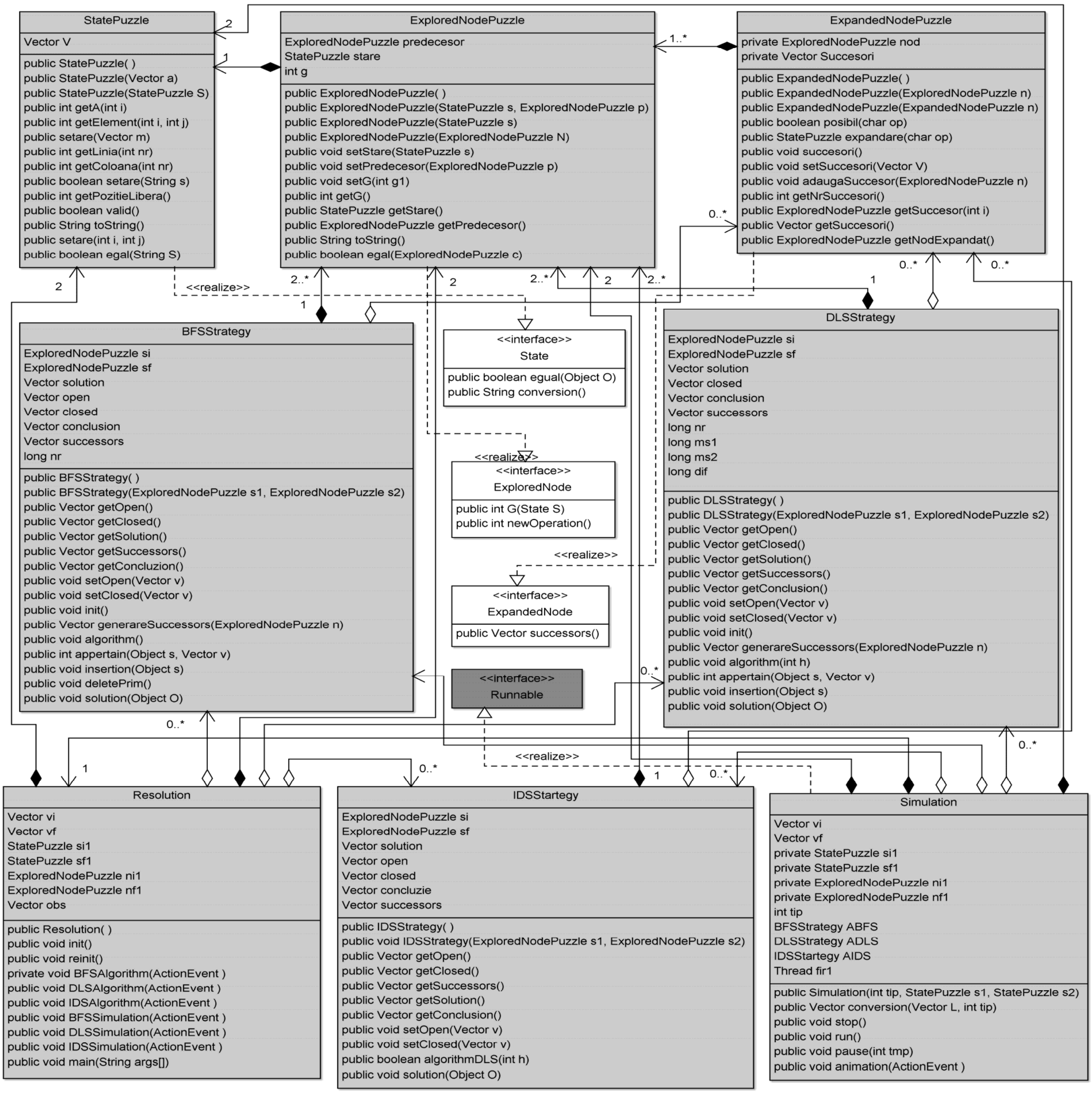

Figure 2. Class Diagram

GRAFICAL INTERFACE OF THE INTERACTIVE SOFTWARE

The interactive software is accomplished using the Java programming language [10]. The application can easily convert in a Java applet. Given that specified requisites in uses cases diagram (figure 1) it was designed graphical user interface of the interactive software.

A configuration on the game table which has to be solved can be introduced by the user or can be automatically generated, is being solved by using the implemented algorithm. In figure 3 is presented the main window of the application, a “Resolution” class instance which contains components that specify the start state, the goal state and an intermediary state of solution, list for viewing the solution obtained by using breadth first search strategy, depth limited search strategy and depth first iterative deepening search strategy, but and also components for comparing the three variances of uninformed search strategies from the point of view of space and time complexity.

In figure 3 are generated the solutions of a configuration which are introduced from the standard input by using the three uninformed search strategies, the optimal solutions is being obtained in 5 steps.

By selecting the button “Simulare BFS” is instancing an object of “Simulation” class which permits simulation of breadth first search strategy. In figure 4 is presented an instance of “Simulation” class, window which permit viewing the way of modifying the open and closed lists for solving a given configuration. The optimal solution for presented example is obtained in 3 steps. For the final step is presented the optimal node from the open list together with its successors. This optimal will be inserted into the closed list after elimination from the open list. Into the open list will be inserted all nodes from successors list that don’t have associated a redundant state. Each node is displayed together with the values of the function used by the strategy: the path cost from the start node to current node G.

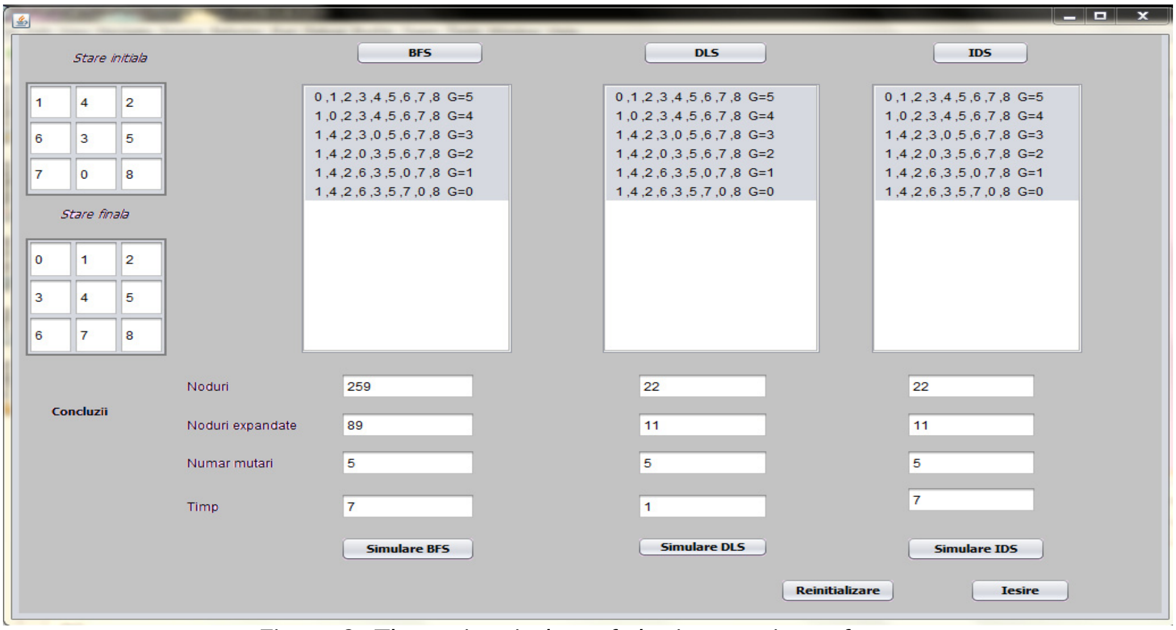

Figure 3. The main window of the interactive software

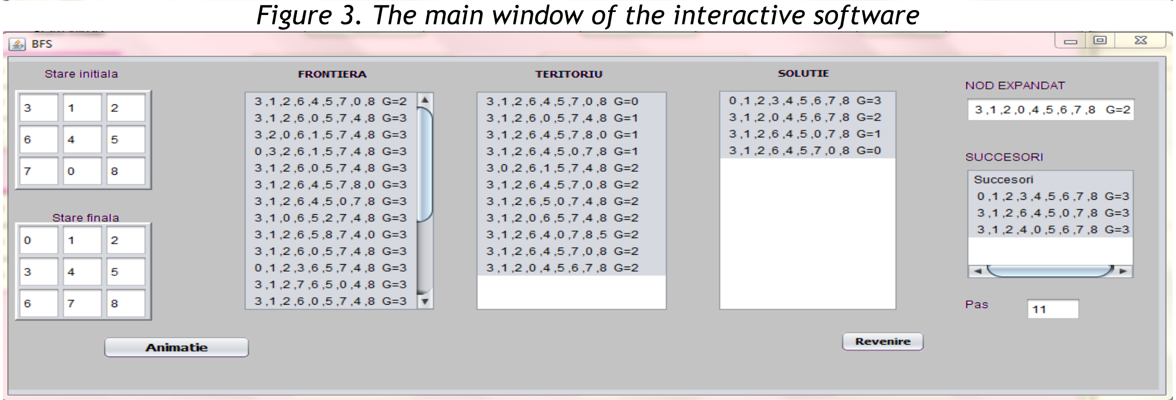

Figure 4. Breadth first search strategy simulation

By selecting the button “Simulare DLS” is instancing an object of “Simulation” class which permits simulation of depth limited search strategy. In figure 5 is presented an instance of “Simulation” class, window which permit viewing the way of modifying the open and closed lists for solving a given configuration. The optimal solution for presented example is obtained in 3 steps.

By selecting the button “Simulare IDS” is instancing an object of “Simulation” class which permits simulation of depth first iterative deepening search strategy. In figure 6 is presented an instance of “Simulation” class, window which permit viewing the way of modifying the open and closed lists for solving a given configuration. The optimal solution for presented example is obtained in 3 steps.

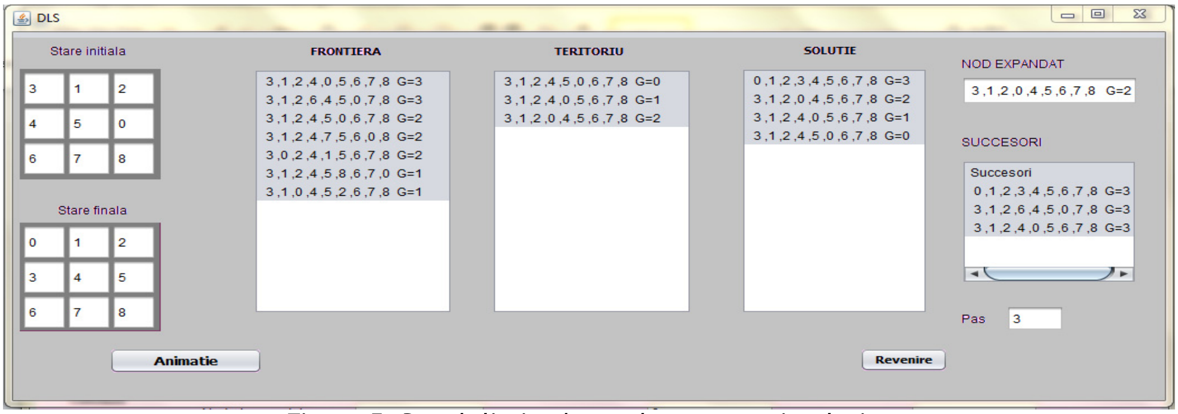

Figure 5. Depth limited search strategy simulation

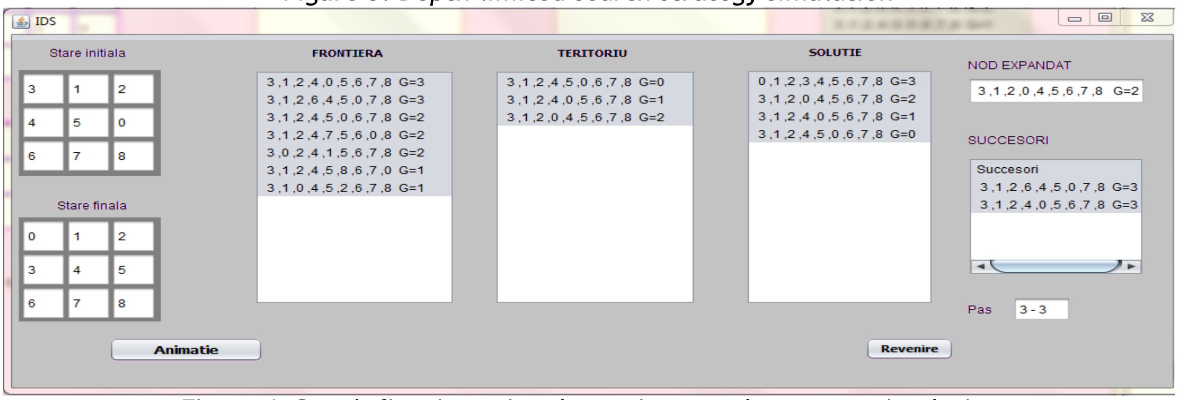

Figure 6. Depth first iterative deepening search strategy simulation

COMPARISON OF EFFICIENCY OF SELECTED STRATEGIES

Because one of the objectives of this paper is to compare the efficiency of selected strategies, the presented results have been obtained using each strategy with the most suitable parameters.

In table 1 are analysed the statistic results regarding the number of the explored nodes by the three uninformed search strategies, in the tables are also presented the minimum number of nodes, the maximum number of nodes and the average value of the explored nodes number which need between 1 and 6 steps. For presenting the efficiency of the depth first iterative deepening search strategy from the explored nodes number point of view there are represented graphical the experimental values obtained for solutions with steps between the interval [1,6] in figure 7.

Table 1. Number of the explored nodes

	Explored Nodes								
	BFS Strategy			DLS Strategy			IDS Strategy		
Number of steps	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum	Average
1.	3	5	4	3	5	4	3	5	4
2.	9	17	10	6	29	18	6	8	7
3.	18	33	26	9	57	43	9	27	17
4.	49	94	82	11	63	33	11	39	24
5.	145	240	192	21	80	43	21	80	43
6.	240	830	519	789	1657	989	32	128	66

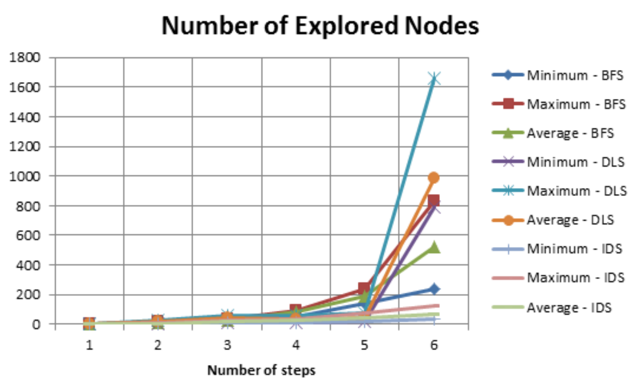

Figure 7. Explored nodes comparing

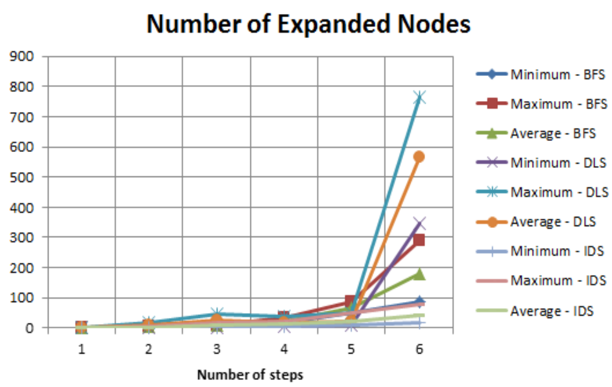

Figure 8. Expanded nodes comparing

Table 2. Number of the expanded nodes

	Expanded Nodes								
	BFS Strategy			DLS Strategy			IDS Strategy		
Number of steps	Minimum	Maximum	Average	Minimum	Maximum	Average	Minimum	Maximum	Average
1.	1	1	1	1	1	1	1	1	1
2.	3	5	4	2	18	10	2	4	3
3.	6	11	8	3	47	26	3	17	9
4.	17	34	24	4	38	19	4	22	12
5.	51	86	65	9	49	24	9	49	24
6.	86	288	180	345	764	564	16	78	41

In table 2 are analysed the statistic results regarding the number of the expanded nodes by the three uninformed search strategies, in the tables are also presented the minimum number of nodes, the maximum number of nodes and the average value of the expanded nodes number which need between 1 and 6 steps.

For presenting the efficiency of the depth first iterative deepening search strategy from the expanded nodes number point of view there are represented graphical the experimental values obtained for solutions with steps between the interval [1,6] in figure 8.

CONCLUSIONS

The main purpose in accomplishing of this study was designing and implementing of didactic interactive software that must lead the user to obtain experience in comprehending and acknowledgment accumulating existing in uninformed search strategies of state space solutions.

From experimentally results by using the depth first iterative deepening search strategy, an improvement in space complexity of uninformed search strategy versus breadth first search strategy and depth limited search strategy was observed. In the three cases the generated solution is obtained using an optimal number of steps.

Theme that is treated into this study is very actually, the artificial intelligence being an important component for forming the young engineers.

REFERENCES

[1] R. Akerkar, Introduction to Artificial Intelligence, Prentice-Hall of India Private Limited, New Delhi, 2005

[2] W. Ertel, Introduction to Artificial Intelligence, Springer-Verlag, Berlin, 2011

[3] E. Kumar, Artificial Intelligence, I K International Private Limited, New Delhi, 2008

[4] Z. Shi, Advanced Artificial Intelligence, Word Scientific Publishing, 2011

[5] M. Hutter, Universal Artificial Intelligence, Springer-Verlag, Berlin, 2005

[6] B. Eckel, Thinking in Java, Prentice Hall, 2003

[7] http://argouml.tigris.org

[8] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison Wesley, 1999

[9] M. Fowler, K. Scott, UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison Wesley, Readings MA, USA, 2000

[10] K. Arnold, J. Gosling, The Java Programming Language, Addison-Wesley, 1998

ANNALS of Faculty Engineering Hunedoara

- International Journal of Engineering