TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 62, Issue I, March, 2019

EFFICIENT METHOD TO SOLVE THE GUARINI PUZZLE GENERALIZATION USING BIPARTITE GRAPHS

Anca-Elena IORDAN, Florin COVACIU

Abstract: *In this paper there it is presented an interactive software implemented in the C# programming language using .Net Frameworks platform which allows a efficient solution of a puzzle obtained through Guarini puzzle generalization. This consists in considering 6 knights placed on the chessboard of 3·n dimension. Three knights are white and are positioned on the first line of the chessboard, and the other three are black and are positioned on the last line of the chessboard. The aim of this puzzle is to move the knights through a minimum number of moves so on the first line we have all the black knights, and on the last line we have all the white knights. This puzzle belongs to a category of problems which can be efficiently resolved using graph theory, that represents a branch of discrete mathematics.*
Key words: *Bipartite graphs, Guarini puzzle, C#, .Net Frameworks, UML*

1. INTRODUCTION

The generalization of Guarini puzzle [1-2] consists in considering 6 knights placed on the chessboard of 3·n dimension: the three white knights are at the first line, and the three black knights are at the last line (figure 1a). The scope of this puzzle is to exchange the knights to obtain the position represented in figure 1b (particular case n=5) in the minimum number of knight movements, not permitting more than one knight on a square at any time.

This puzzle belongs to a class of problems which can be resolved using different types of algorithms corresponding to areas such as discrete mathematics (graph theory) or artificial intelligence (heuristic methods, neural networks [3]).

In this paper it is presented an efficient method to solve the puzzle by using bipartite graphs [4]. The process of modeling the problem through the bipartite graph consists in numbering the squares of the chessboard which will represent the vertices of the bipartite graph. The possible movements of the knights on the chessboard, which look like the letter "L", will represent the edges of the bipartite graph.

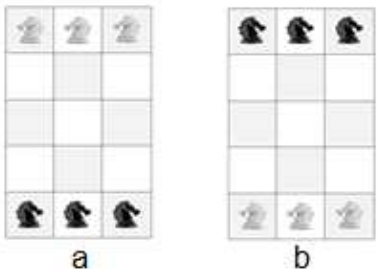

Fig. 1. a) Initial state b) Final state

2. BIPARTITE GRAPHS

In graph theory a multipartite graph is a graph whose vertices can be divided into m different independent sets [5]. In other words, it is a graph that can be colored with m colors, so that no two endpoints of an edge have the same color. When m = 2 these are the bipartite graphs [6] (figure 2a).

A complete multipartite graph [7-8] is a multipartite graph in which there is an edge between every pair of vertices from different independent sets. These graphs are described by notation with a capital letter K subscripted by a sequence of the sizes of each set in the partition [9]. For example, in figure 2b it is presented the complete bipartite graph $K_{4,4}$.

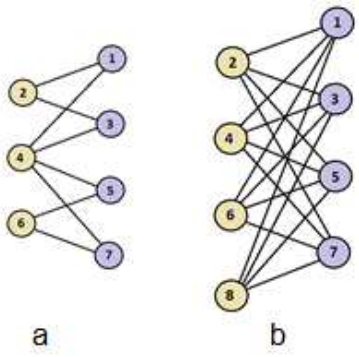

Fig. 2. a) Example of a bipartite graph b) Complete bipartite graph $K_{4,4}$

3. ANALYSIS STAGE IN SOFTWARE DEVELOPMENT

The scope of this paper is to present a software which permits an optimal solution of the puzzle presented in the first paragraph. The mathematical modeling of this puzzle with the aim of an optimal resolving can be used to teach bipartite graph notion [10].

From the perspective of unified modelling language [11], the analysis of the interactive software includes the representation of the use-case diagram and of the activity diagrams [12]. The use-case diagram offers simplified and graphical representation of what the software must really do. Use-case diagram (figure 3) is based upon functionality and thus will focus on the "what" offers the interactive software and not how will be realized.

The use-case diagram corresponding to this software includes:

- One actor - the user who is external entity with that the software interacts;
- Four use-cases which describe the performance of the software;
- Association relationships between player and use-cases, and dependency relationships between use-cases.

Also, in the analysis stage of the interactive software, one activity diagram it is realizing for each use case present in the diagram from figure 3. Figure 4 shows the activity diagram corresponding to the "Generation of the bipartite graph" use-case that includes seven activities. For each activity will be implemented an algorithm having final goal the graphical representation of the bipartite graph associated with the puzzle.

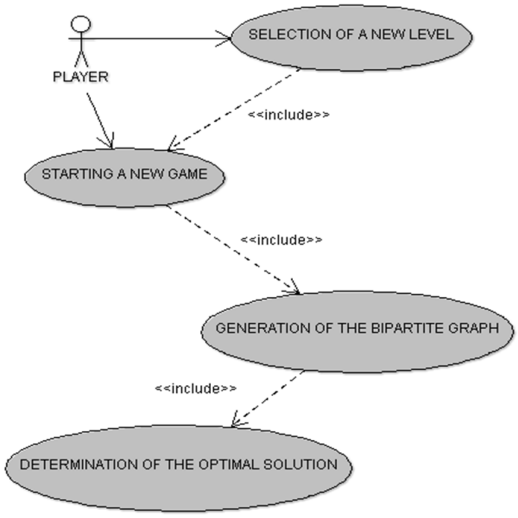

Fig. 3. Use-case diagram

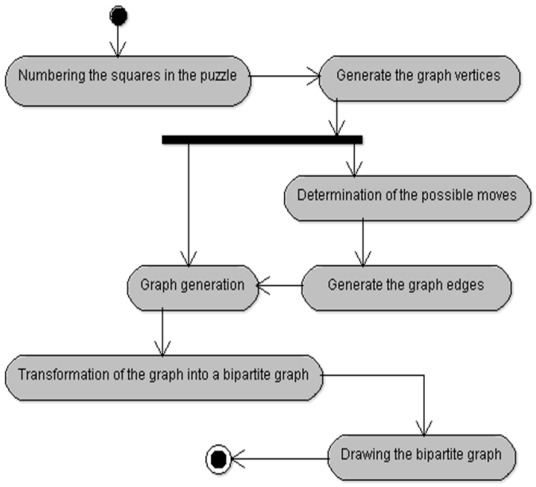

Fig. 4. Activity diagram

4. DESIGN STAGE IN SOFTWARE DEVELOPMENT

Conceptual modeling allows identifying the very important elements for the interactive software. Class diagram [13] is represented 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.

After identifying the specific elements of the interactive software, they were been implemented 3 interfaces and 9 classes grouped into three packages.

Package "GameState" brings together concepts that correspond to the algorithm corresponding to the bipartite graph, consisting of 2 classes and 2 interfaces (figure 5). For memorizing a state configuration, there was implemented "StateGame" class which implements "IStateGame" interface. The algorithm corresponding to bipartite graph is implemented by using "Algorithm" class which implements "IAlgorithm" interface. An instance of this class is composed by one instance of "StateGame" class and one instance of "BipartiteGraph", according to composition relationship which exists in diagram.

Package "GraphicalUserInterface" brings together concepts that correspond to the graphical user interface corresponding to the interactive software, consisting of 3 classes (figure 6). For designing the graphical interface of the software, it was implemented the „MainInterface" class. An instance of this class is composed by one instance of "BipartiteClassRepresentation" class and one instance of "StateGameRepresentation" class, according to composition relationship which exists in diagram. The "StateGameRepresentation" class was implemented for show the current state of the puzzle, and the "BipartiteGraphRepresentation" class was implemented to present the bipartite graph associated to the current state of the puzzle.

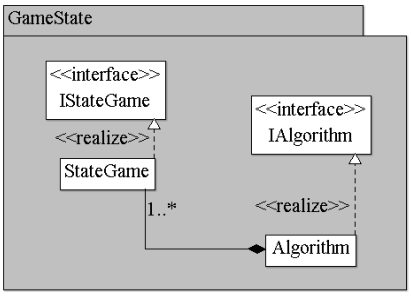

Fig. 5. The structure of "GameState" package

Package "Graph" (figure 7) brings together concepts that correspond to the graph used to solve the puzzle, consisting of 4 classes and one interface. The concept of a vertex from a graph is implemented by using "Vertex" class which implements "GraphicalRepresentation" interface.

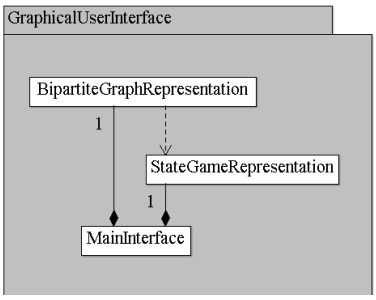

Fig. 6. The structure of "GraphicalUserInterface" package

The concept of an edge from a graph is implemented by using "Edge" class which implements "GraphicalRepresentation" interface. An instance of this class is composed by two instance of "Vertex" class.

The concept of graph is implemented by using "Graph" class which implements "GraphicalRepresentation" interface. An instance of this class is composed by several instances of "Vertex" class and several instances of "Edge" class, according to composition relationship which exists in package diagram.

The concept of bipartite graph is implemented by using "BipartiteGraph" class which inherits from "Graph" class. In the design phase, it also performs statechart diagrams that specify the dynamic behavior of class instances. Figure 8 shows the statechart diagram associated with an instance of the "Algorithm" class.

Also, collaboration diagrams [14] are developed in the design stage with the aim to show the interactions between objects to solve the use-cases presented in figure 3. Figure 9 shows the collaboration diagram that includes the interactions between 8 types of objects that allow drawing of the bipartite graph.

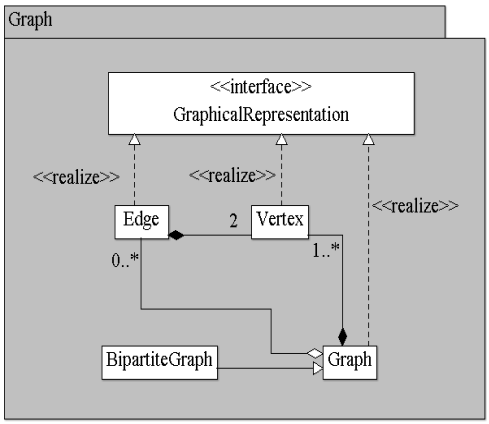

Fig. 7. The structure of "GameState" package

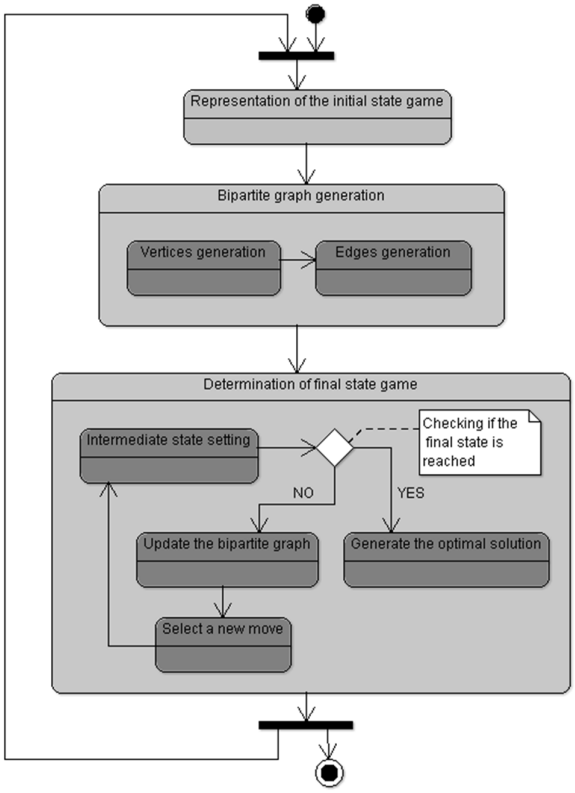

Fig. 8. Statechart diagram

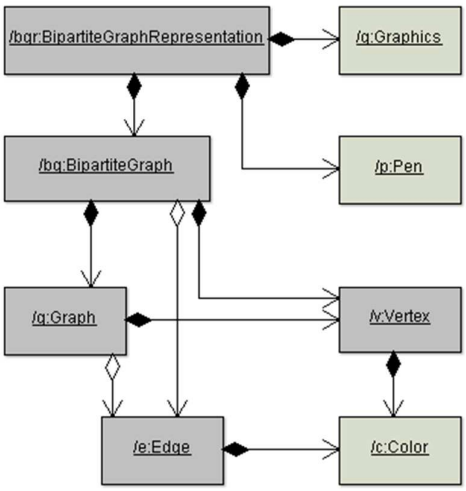

Fig. 9. Collaboration diagram

5. IMPLEMENTATION STAGE IN SOFTWARE DEVELOPMENT

Using UML, the implementation stage consists in a deployment diagram that describes the collection of components that ensure the functionality of the application [15]. In this type of diagram (figure 10) it can see how the software is divided, but also the dependencies between the 12 modules.

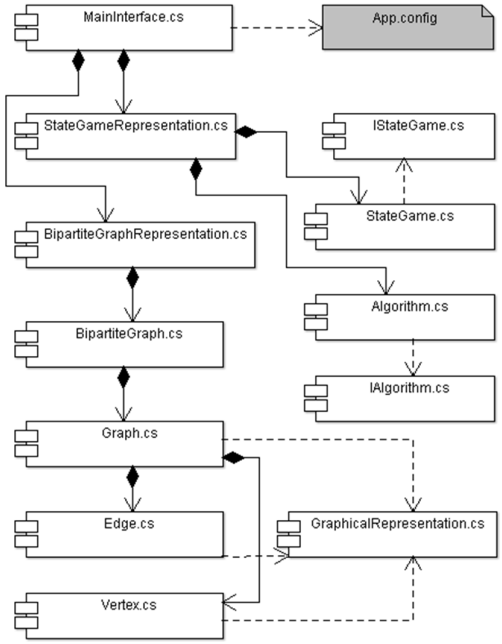

Fig. 10. Deployment diagram

6. GRAPHICAL USER INTERFACE

The interactive software is accomplished using the C# object-oriented programming language [16-17] on the .Net Framework platform [18]. Given that specified requisites in use-case diagram (figure 3) it was designed graphical user interface.

After the level of difficulty was selected, it is generated the bipartite graph associated to the start state (figure 11), and the player can begin the game. The application permits movements of any knight between two adjacent vertices, such that any vertex of the bipartite graph does not contain more than one knight at any time.

After each movement, the current state of the puzzle is updated (figure 12) and it is checked whether the final state (figure 13) has been reached.

7. CONCLUSION

In this paper the graph theory, which is a branch of discrete mathematics, was applied to the game theory to efficiently solve the puzzle presented in figure 1. Because the graph theory is a difficult field of discrete mathematics, the computer games can be of helpful in the

teaching/learning process [19] of manifold aspects regrading to graphs.

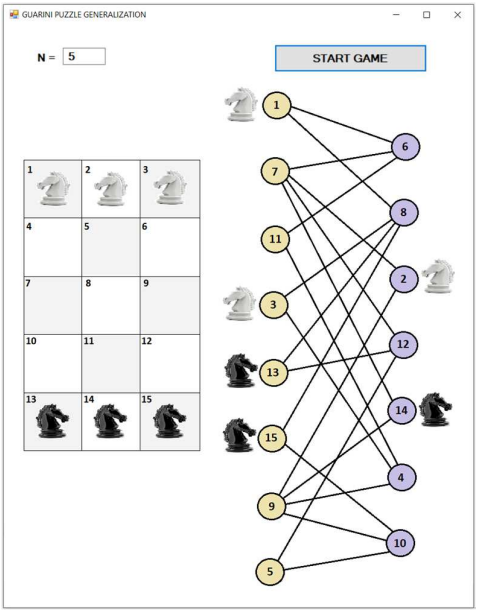

Fig.11. Graphical user interface corresponding to the initial state of the 6 knights puzzle

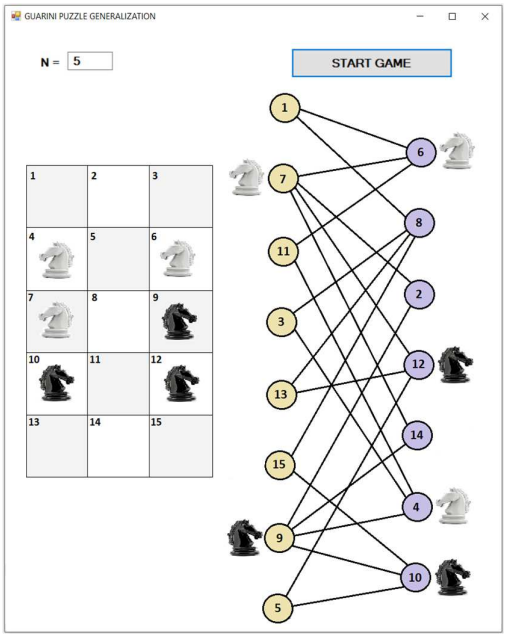

Fig.12. Graphical user interface corresponding to the intermediate state

The utilization of interactive software signifies an efficient way of stimulation and development of the student motivation.

The interactive software presented in this paper contributes to the qualitative assimilation of the knowledge and to the competences development related to the bipartite graphs, having great educational opportunities.

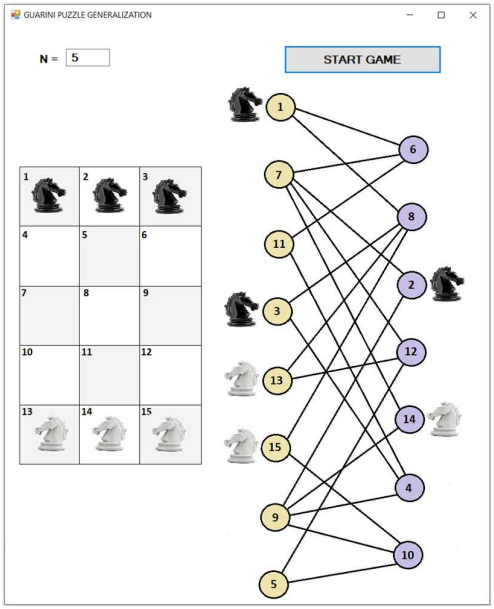

Fig.13. Graphical user interface corresponding to the final state of the 6 knights puzzle

8. REFERENCES

[1] Levitin, A., Levitin, M., *Algorithmic Puzzles*, Oxford University Press, ISBN: 978-0-19-974044-4, New York, USA, 2011

[2] Devadas, S., *Programming for the Puzzled: Learn to Program while Solving Puzzles*, MIT Press, ISBN: 9780262534307, Cambridge, MA, USA, 2017

[3] Barabás, I., Todoruţ, A., Cordoş, N., *An Artificial Neural Network Approach to Estimate the Viscosity of Biodiesel-Diesel-Ethanol Blends*, Acta Technica Napocensis, Series: Applied Mathematics, Mechanics, and Engineering, no. 59 (3), pp. 245- 250, ISSN: 1221 – 5872, 2016

[4] Asratian, A., Denlev, T., Haggkvist, R., *Bipartite Graphs and their Applications*, Cambridge University Press, ISBN: 9780511984068, Cambridge, United Kingdom, 2011

[5] Chartrand, G., Zhang, P., *Chromatic Graph Theory*, CRC Press, ISBN: 978-1-58488-801-7, Boca Raton, Florida, USA, 2009

[6] Pena, J., Rochat, Y., *Bipartite Graphs as Models of Population Structures in Evolutionary Multiplayer Games,* PLoS ONE 7(9), ISSN: 1932-6203, 2012

[7] Allagan, J., Serkan, C., *Bell Numbers of Complete Multipartite Graphs*, Computer Science Journal of Moldova, vol.24, no.2(71), pp. 234-242, ISSN: 1561-4042, 2016

[8] Dobocan, C.A., Blebea, I., Popescu. D., *A Mathematical Model Applied to an Economical Process*, Acta Technica Napocensis, Series: Applied Mathematics, Mechanics, and Engineering, no. 56(1), pp. 115- 120, ISSN: 1221 – 5872, 2013

[9] Gethner, E., Hogben, L., Lidický, B., Pfender, F., Ruiz, A., Young, M., *On Crossing Numbers of Complete Tripartite and Balanced Complete Multipartite* Graphs, Journal of Graph Theory, vol. 84(4), pp. 552-565, ISSN: 1097-0118, 2017

[10] Wang, G., Liu, G., *Rainbow Matchings in Properly Colored Bipartite Graphs*, Open Journal of Discrete Mathematics, vol. 2, pp. 62-64, ISSN: 2161-7635, 2012

[11] Bruegge, B., Dutoit, A., *Object Oriented Software Engineering Using UML*, Prentice-Hall, Inc. Upper Saddle River, ISBN: 978-0-13-606125-0, NJ, USA, 2013

[12] Dennis, A., Wixom, B., Tegarden, D., *Systems Analysis and Design with UML*, John Wiley & Sons Ltd, Hoboken, NJ, USA, 2012

[13] Rumpe, B., *Modeling with UML*, Springer International Publishing AG, ISBN: 978-3-319-33932-0, Switzerland, 2016

[14] Lincke, S., Knautz, T., *Designing System Security with UML Misuse Deployment Diagrams*, Proceedings of the IEEE Sixth International Conference on Software Security and Reliability Companion, pp. 57-61, ISBN: 978-0-7695-4743-5, Gaithersburg, Maryland, USA, June 2012

[15] Gaşpar, M.L., Firescu, V., *New Skills and Qualifications Required by the Current Approaches in the Software Development Industry*, Acta Technica Napocensis, Series: Applied Mathematics, Mechanics, and Engineering, no. 61, pp. 97- 106, ISSN: 1221 – 5872, 2018

[16] Doyle, B., *C# Programming From Problem Analysis to Program Design*, Cengage Learning, ISBN: 978-1285096261, Australia, 2013

[17] Purdum, J., *Beginning Object Oriented Programming with C#*, John Wiley & Sons, Inc., ISBN: 978-1118336922, Indianapolis, Indiana, USA, 2012

[18] Thuan, T., Hoang, L., *.Net Framework Essentials. Introducing the .NET Framework*, O'Reilly & Associates, Inc. Sebastopol, ISBN: 978-0596005054, CA, USA, 2015

[19] Popescu, D.I., *Teaching Computer Aided Design for Engineering Students*, Acta Technica Napocensis, Series: Applied Mathematics, Mechanics, and Engineering, no. 58(3), pp. 331- 336, ISSN: 1221 – 5872, 2015

METODĂ EFICIENTĂ DE REZOLVARE A UNEI GENERALIZĂRI A PUZZLE-ULUI GUARINI UTILIZÂND GRAFURI BIPARTITE

Rezumat: În această lucrare este prezentat un soft interactiv implementat în limbajul de programare C# utilizând platforma .Net Frameworks care va permite o rezolvare eficientă a unui puzzle obținut prin generalizarea puzzlel-ului Guarini. Acesta constă în considerarea a 6 cai plasați pe o tablă de șah de dimesiune 3·n. Trei cai sunt albi și sunt plasați pe prima linie de pe tabla de șah, iar ceilalți trei sunt negri și sunt plasați pe ultima linie de pe tabla de șah. Scopul acestui puzzle constă în a muta caii printr-un număr minim de mutări astfel încât pe prima linie să fie toți caii negri, iar pe ultima linie toți caii albi. Acest puzzle aparține unei categorii de probleme care poate fi rezolvată eficient utilizând teoria grafurilor, ce reprezintă o ramură a matematicii discrete.

Anca-Elena IORDAN, PhD, Lecturer, Polytechnic University of Timisoara, Department of Electrical Engineering and Industrial Informatics, anca.iordan@fih.upt.ro, Revolutiei 5 Street, Hunedoara, 331128, ROMANIA, +40724578986

Florin COVACIU, PhD Eng., Lecturer, Technical University of Cluj-Napoca, Department of Design Engineering and Robotics, florin.covaciu@muri.utcluj.ro, Blvd. Muncii 103-105, Cluj-Napoca, 400641, ROMANIA, +40755566491