Gamification of Computational Problems

End-Semester Report of
5th Semester Mini Project
FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

BY

Milind Chauhan (IIT2014117)
Suresh Kumar Dhayal (IIT2014060)
Navratna Sharma (IIT2013148)

UNDER THE SUPERVISION OF

Dr. Rishi Ranjan Singh
Asst. Professor
IIIT-ALLAHABAD

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

NOVEMBER, 2016

CANDIDATES' DECLARATION

We hereby declare that the work presented in this project report entitled "**Gamification of Computational Problems**", submitted end-semester report of 5th Semester of B.Tech. (IT) at Indian Institute of Information Technology, Allahabad, is an authenticated record of our original work carried out from Aug 2016 to Dec 2016 under the guidance of **Asst. Prof. Rishi Ranjan**. Due acknowledgements has been made in the text to all other material used. The project was done in full compliance with the requirements and constraints of the prescribed curriculum.

Place: Allahabad
Date:

Milind Chauhan (IIT2014117)
Suresh Kumar Dhayal (IIT2014060)
Navratna Sharma (IIT2013148)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Place: Allahabad

Dr. Rishi Ranjan Singh
Asst. Professor
IIIT-Allahabad

1. Introduction

The earliest known video game was written by Alan Turing and David Champernowne in 1948. It was called *Turochamp* and it was a chess simulation. It was never implemented though. The first games to be actually implemented were the machines *Bertie the Bot* [2] and *Nimrod* [6]. They played the game of *Tic Tac Toe* and *Nim* respectively. But back then games were just pieces of code to exhibit the machines capabilities. Both the games and the machines that they are played on have come a long way since then. In present times the machines have become more mobile and accessible. Smartphones, tablets and PCs, all incredibly powerful and cheap. They have become a necessary part of our lives due to their usefulness and daily applications. One such application of these machines is playing games.

In recent years, many of these games have been proven to be NP-Hard. For eg:

- Super Mario Bros [7],
- Some problems related to Tetris [8],
- Candy Crush Saga [1] etc.

This revelation that the computational problems behind a game can be NP-Hard might shed some light on questions such as:

- Why are some levels of a game harder than others?
- Given a particular instance (level) of a game, is it always possible to win (achieve the goal) that level? Etc.

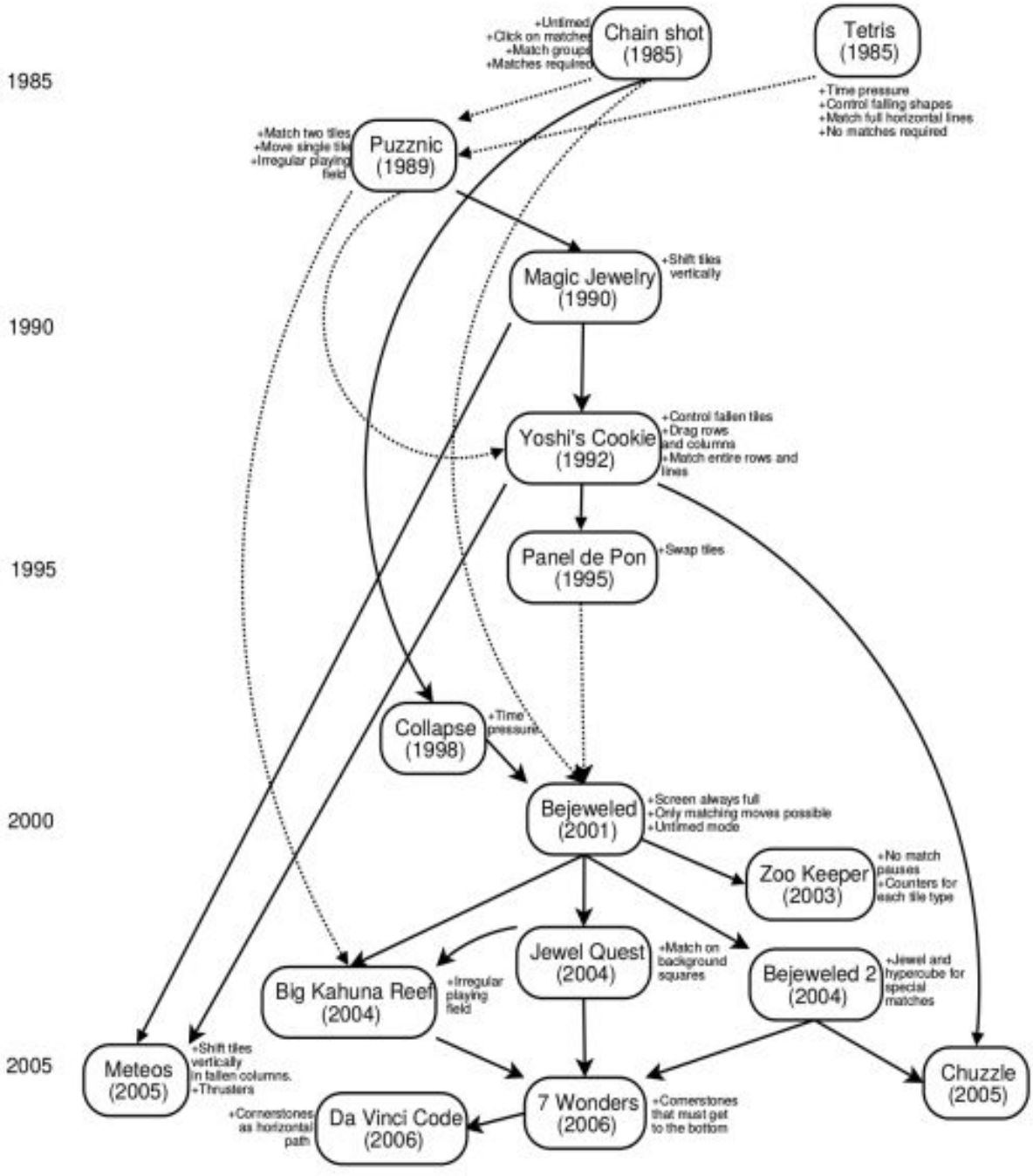


Figure 1: The “genealogy” of Bejeweled[12]

2. Problem Definition and Objectives

1. How many existing games are NP-Hard?

   There already exists a large number of games that are proven to be NP-Hard.

   - Candy Crush Saga [1]
   - Donkey Kong [7]
   - Problems related to Tetris [8]
   - Metroid [7]
   - (Generalized) FreeCell [9]
   - Legend of Zelda [7]
   - Minesweeper Consistency Problem [10]
   - Pokémon [7]
   - Super Mario Bros [7]

And there are many more. Our objective is to study the analysis of some of the existing games and try to understand and learn the process that is required in proving the questions related to these games to be NP-Hard.

2. To Learn the process of proving a problem to be NP-Hard and implement a new game based on an NP-Hard problem, which is some ways in the future. Therefore, our two main objectives will be to:

   1. Analyze the relationship between games and computational problems
   2. Learn the methods, constructions and processes required to prove a game to be NP-Hard
   3. Implement a new game based on an NP-Hard Problem.

3. Literature Survey

1. Introduction To Algorithms, by Thomas Cormen et al.[11]

   a.) P Class of problems: The P class contains those computational problems that are solvable in polynomial time. i.e. There are polynomial time algorithms for solving these problems. For eg: the sorting problem, which is in class P

   b.) NP Class of problems: The NP class of problems are those problems that are verifiable in polynomial time. i.e. Given a specific solution to the problem, one should be able to check in polynomial time whether it's correct or not. Also, P is a subset of NP, since if you can solve a problem in polynomial time, you can also verify it in polynomial time.

   c.) NP-Hard: A problem is NP-Hard if it is at least as 'hard' as all the other problems in NP. It means that if there exists a polynomial time algorithm to solve a problem that is NP-Hard, then there exists polynomial time algorithms for all the problems that are NP. The important distinction between NP and NP-Hard is that a NP-Hard problem need not always be verifiable in polynomial time.

2. Introduction to the Theory of Computation, by Michael Sipser[5]

   a.) Reducibility: A problem *A* is said to be reducible to problem *B* if given an instance of *A* there is polynomial time process that transforms it into an instance of *B*. This is called polynomial time reducibility.

   b.) If *A* can be efficiently (in polynomial time) reduced into *B* then if there is an efficient (polynomial time algorithm) way to solve *A,* then *B* can also be solved efficiently.

   c.) Cook-Levin Theorem: *SAT is NP-Complete*

- *SAT* is the Boolean ***satisfiability problem.***
- Description of the ***satisfiability problem***: Given a Boolean formula in CNF (Conjunctive Normal Form), is there an assignment of the literals (setting them to TRUE or FALSE) that results in the Boolean formula to evaluate to TRUE.
- A Boolean formula is said to be CNF if it is conjunction of clauses where the clauses are a disjunction of literals.
- Examples of Boolean formulas in CNF:
  - $\neg A \wedge (B \vee C)$
  - $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$

d.) How to prove a problem *B* to be NP-Complete:

- Reduce an NP-Complete problem *A* to *B*.
- Now let's assume that there is an efficient way to solve to *B*. If there is an efficient way to solve *B* then there has to be an efficient way to solve *A*, since *A* had been reduced to *B*. But that can't be as *A* is NP-Complete. Hence our assumption that there is an efficient way to solve *B* is wrong and *B* is NP-Complete.
- Hence, to prove a problem *B* to be NP-Complete, reduce an NP-Complete problem to *B*.

3. Bejeweled, Candy Crush and other Match-Three Games are NP-Hard, by L. Gualà et al.[3]
   a. Bejeweled is a popular *tile matching match-3* type game where the player has to swap gems to form matches of three similar gems. The matched group then deletes itself and the gems above it fall down to fill in the empty space.
   b. Gameplay of Bejeweled:
      - Setup: An 8 x 8 grid on a board filled with six different types of gems.
      - A gem is adjacent to the gems directly to the North, South, East and West to it.

- A legal move: The player can swap a gem with an adjacent one if the swap results in an either vertical or horizontal alignment of three similar gems.
- The matched gems are then ‘popped’ or deleted. The gems above them fall down to fill in the empty places and new gems are generated at the top of the board.
- If a swap results in more than one matched group, then all the matched groups are popped simultaneously together.
- This process continues until there are no legal moves left to make.

c. Reduction Overview:
   - Reduction will be of an instance of the *1-in-3 positive SAT* (*1in3PSAT*) which is NP-Hard [11], where the goal is to satisfy the Boolean formula such that there is exactly one literal set to TRUE in every clause to an instance of the generalized Bejeweled game with a board of dimensions w x h where the goal is to pop a specific gem at a specific location. The game has six different gems which we will denote by A, B, C, D, E and F.

d. There are a total of 6 gadgets; the sequencer, the choice activator, the choice wire, the displacer, the automatic wires and the goal wire, that will be arranged in the board like this:

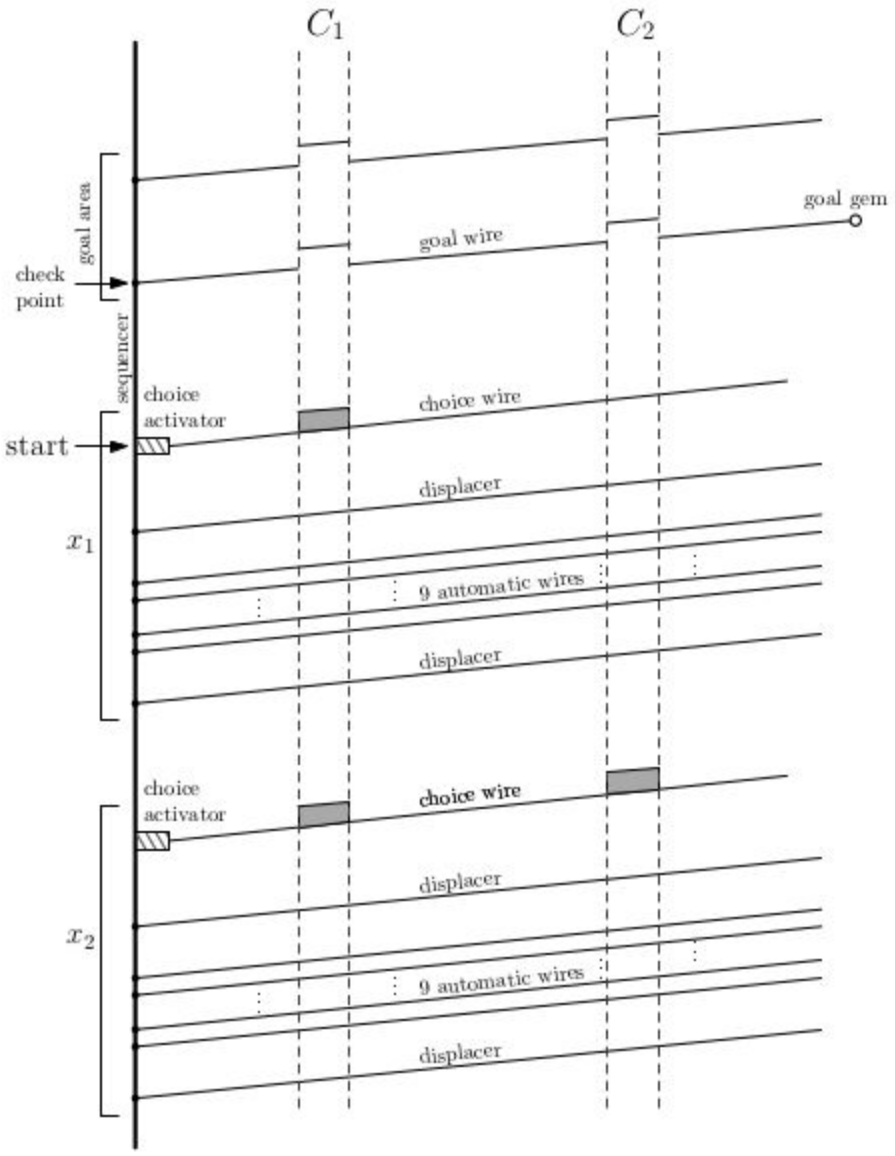


Figure 3: Overview of the structure of the generated instance.

e. The first column in its entirety is the sequencer gadget. It has two main functions:
   - Passing the control from one variable to next.
   - Activating the displacers and the automatic wires

Figure 4: The Sequencer gadget

- In Figure 4, the parts in red are the parts that will activate the wires by coming in contact with them and forming a match and thus popping out. The parts in grey are example gadgets that do nothing except interact with the sequencer.

f. Whenever you set the choice activator for a variable to TRUE, the choice wire is set in such a way that it get it's popped out in its entirety via a set of consecutive matches.

Figure 5: Choice Activator gadget

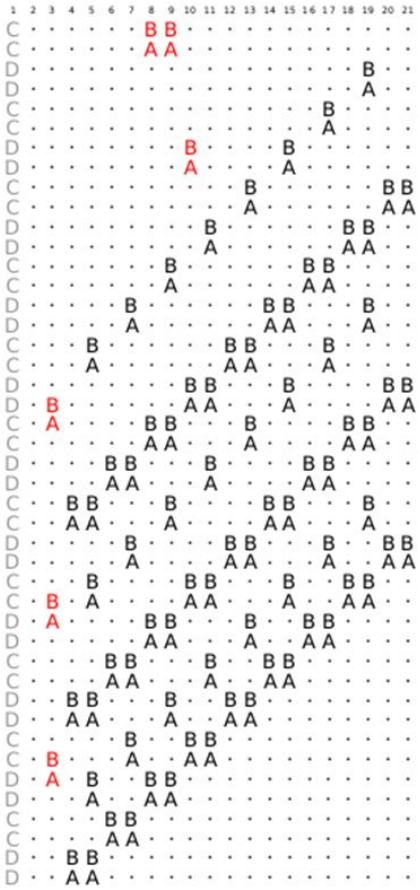


Figure 6: Choice Wire gadget

- Each clause is represented by a set of three consecutive columns the locations of which are represented by the formula: (2*j + 6, 2*j + 8), where j = 1 for the first clause and so on.
- The gadgets ensure that all rows fall by a number that is a multiple of six, while the clauses columns have extra gems for the literals they contain. If the clause is satisfied by the assignment of its literals, those extra gems match up to make the clause columns fall by 2 extra rows and line up with the goal line.

- If all the clauses are satisfied then the goal line becomes complete and then it is easy to pop the goal gem.
- Hence an instance of *1-in-3 positive SAT problem* is reduced to an instance of the bejeweled problem of popping a specific gem.

4. The Graph Coloring Problem
   It is an NP-Complete problem to check if a graph accepts a K-coloring for a given K except for K equal to 0,1 or 2, such that no adjacent vertices have the same color. It's also an NP-Hard problem to compute the chromatic number[4]

4. Hardware and Software Requirements

1. Sublime-text
2. Chrome Browser
3. HTML (Hyper Text Markup Language)
4. JavaScript
5. CSS (Cascading Style Sheet)
6. Color Picker

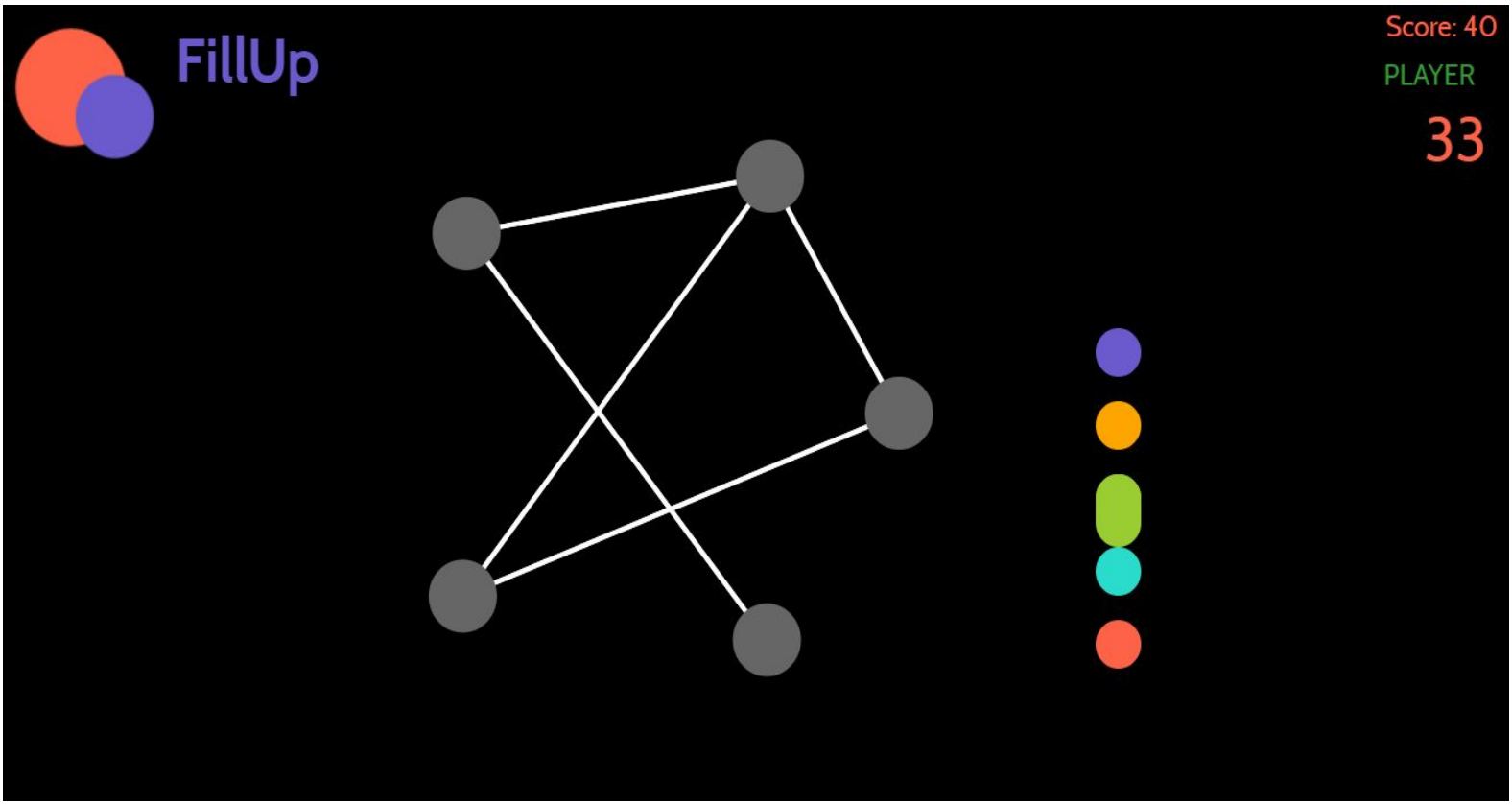
5. Results and Analysis

We successfully developed a new game based on an NP-Complete problem. The NP-Complete problem in questions is the graph coloring problem. Given a graph it is an NP-Complete problem to determine if a graph accepts a k-coloring.[4]

Design Principles of the Game:

- The player is started off with a randomly generated graph and five colors which they then have to use to color the vertices of the graph.
- The player cannot assign a color to a vertex if any adjacent vertex has the same color.
- The player has to fill all the vertices with the color within the given time.
- If the player succeeds in doing so, they may advance to the next level.
- The score of the player depends on the amount of colors they use, with usage of 3 colors resulting in the highest score and usage of 5 resulting in the lowest
- Each successive level adds more vertices and edges to the graph to make the level harder.
- The objective of the user is to reach the highest level.

Snapshots of the Game:



References

[1] Toby Walsh. “Candy crush is np-hard”, arXiv preprint arXiv:1403.1911, 2014

[2] Chris Bateman. "Meet Bertie the Brain, the world's first arcade game, built in Toronto". Spacing Toronto. August 13, 2014

[3] L. Gualà; S. Leucci; E. Natale (2014). "Bejeweled, Candy Crush and other Match-Three Games are NP-Hard". arXiv:1403.5830

[4]Garey, M. R.; Johnson, D. S.; Stockmeyer, L. (1974), "Some simplified NP-complete problems", Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, pp. 47–63, doi:10.1145/800119.803884

[5]M. Sipser, *Introduction to the theory of computation*, 1st ed. Boston: PWS Pub. Co., 1997.

[6] Tristan Donovan. “Replay: The History of Video Games”. East Sussex: Yellow Ant. ISBN 978-0956507204, 2010

[7] G. Aloupis; E. D. Demaine; A. Guo. "Classic Nintendo Games are (NP-)Hard". arXiv:1203.1895, 3 sept 2012.

[8] Eric D. Demaine; Susan Hohenberger; David Liben-Nowell. Tetris is Hard, Even to Approximate. Proceedings of the 9th International Computing and Combinatorics Conference (COCOON 2003). Big Sky, Montana, July 25–28, 2003

[9] Malte Helmert, Complexity results for standard benchmark domains in planning, Artificial Intelligence Journal 143(2):219-262, 2003

[10] Richard Kaye,"Minesweeper is NP-complete". Mathematical Intelligencer. 22 (2): 9–15. doi:10.1007/BF03025367

[11]T. Cormen, C. Leiserson, C. Stein and R. Rivest, *Introduction to algorithms*, 1st ed. Cambridge, Mass.: MIT Press, 2009.

[12] Juul, Jesper. "Swap Adjacent Gems to Make Sets of Three: A History of Matching Tile Games". Artifact journal. Volume 2, 2007. London: Routledge