Introduction

In Computer Science, Graph isomorphism problem is the problem of determining whether two finite graphs are isomorphic or not. If two graphs contain the same number of graph vertices connected in the same way, then they are said to be isomorphic. The exact graph matching problem is used in a variety of different pattern recognition contexts. In fact, graphs are used to support structural descriptions as well as for low level image representations.

This problem is of special interest in Computational Complexity Theory as it is one of a very small number of problems belonging to NP(Nondeterministic Polynomial Time) neither known to be solvable in polynomial time nor NP-Complete[2] where computational complexity of a problem is the amount of resources, such as time or space, required by a machine to solve the problem. This being the case, it is extremely difficult to design an efficient algorithm to solve this problem in a reasonable amount of time. The time requirements of brute force matching algorithms increase exponentially with the size of the input graphs, restricting the applicability of graph based techniques. This is where we can employ Human Computing techniques to solve this problem in an efficient manner.

Ehrenfeucht-Frasse (EF) games were first developed in the 50's and 60's by Andrzej Ehrenfeucht and Roland Frasse[3]. EF-games are interesting, easy to understand, and very useful
for determining if two structures are logically equivalent or not. There is an implementation of the Ehrenfeucht-Fraisse Game in Java called efGraph[1]. But it does not have a good user interface as ours and does not scale well for deployment on the web. Also it does not contain any option for the developer to specify the graphs he wishes to test for isomorphism.

METHODS

To model the graph isomorphism problem, we tried to implement a simple GWAP (Game with a Purpose). This was implemented using Unity3D[4] which is a popular game development software. Unity3D allows us to create games that can be deployed on a variety of platforms.

The input to our game are two text files that represent the two graphs that need to be tested for isomorphism. The graphs must be given in a predefined format where the first line should contain the word "NODE". The next line contains the number of nodes followed by the edges which are represented as a pair of nodes separated by a tab space in between them. For example, a complete specification of a graph would look like this :

```
Node
3
0	1
1	2
2	0
```

Note that this specification is for undirected graphs.

The output is determined by the player who wins. If the DUPLICATOR wins, the two graphs are said to be isomorphic as he has successfully copied every node in both the graphs. Conversely, if the SPOILER wins, the the graphs are not isomorphic.

Since we have implemented the graph isomorphism problem as a game with a purpose, it requires a human to determine the solution. The human interactions with our system are as follows.

INSTRUCTIONS TO PLAY THE GAME

The game generates two graphs from an external file that is not accessible by the player but be can be modified by the developer. The game consists of two players (SPOILER and DUPLICATOR taking alternate turns) or one player playing both the roles alternatively. On starting the game, the Player is presented with two graphs. SPOILER begins the game by selecting any node from either of the graphs. The DUPLICATOR then has to select a corresponding node from the other graph such that it is similar to the node that SPOILER has selected. The aim of the DUPLICATOR is to copy every move of the SPOILER. The SPOILER tries to choose nodes that the DUPLICATOR cannot copy. The game continues to a predefined number of rounds where the DUPLICATOR tries to mimic every round of the SPOILER. If at the end of the rounds, DUPLICATOR is able to mimic every move, then he wins. Else, SPOILER is declared to be the winner.

RESULTS

- For simple graphs our game worked well giving the desired results. We were unable to test it on a large scale due to time constraints and would like to consider it as a future extension to our project to see how it works for with complicated structures.
- Currently, the game is a prototype as a Unity package and further development needs to be done in order for it to be fully deployable.
- Since we had very little resources available to help us in our implementation of the EF Game, we had to manually deduce the logic and code the algorithm from scratch

FUTURE WORK

- Currently, our game reads only a single file to make the graphs.We would like it to read from a database of files all containing graph specifications.
- Once this has been implemented, we can introduce the concept of Levels where with

increasing Levels, more complicated graphs can be given.
- The number of Levels now are static and predetermined. We would like to modify the Game to change the number of Levels according to the complexity of the graph.
- The graphics can be made more interactive to make a 3D implementation of the game that real users will play.
- A study could be carried out on how the number of rounds and the placement or number of nodes in a graph affect player performance and their results could be incorporated to improve the game and make it more appealing to the players.

DISCUSSION

Humans are able to view big graphs as a an abstract entity and recognize patterns in them as opposed to computers which process information node by node through each iteration. In other words, humans are able to see the big picture and recognize patterns in them.

The game that we have created can be deployed on android devices and various web browsers to be played. Lots of interesting graphs can be constructed and tested for isomorphism in this way. There is also an exciting possibility of deploying this game on the Amazon Mechanical Turk and inviting the Turkers to play the game to evaluate its performance and efficiency. In order to deploy it on the Turk, we would need to add more graphs that the player can choose from. Also, the complexity of the graphs could increase with the increase in the number of rounds thereby increasing the complexity level of the game.

Even though the EF game is a very old approach to solving the isomorphism problem, it is difficult to see its actual implementation. It mostly remains as a mathematical concept. By actually creating a game from it, we are providing the opportunity to test several interesting winning strategies that have so far been researched only theoretically. The game provides a framework upon which the concepts of Ef game can be tested for viability.

Earlier attempts to solve the isomorphism problem was by implementing Tower Defense games. We found during our project that it can also be solved by implementing the Ef game which is an efficient way to harness the human intellect to solve a challenging problem in this field.

References

[1] https://code.google.com/p/efgraph/
[2] Garey, Michael R.; Johnson, David S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, ISBN 978-0-7167-1045-5, OCLC 11745039
[3] A Friendly Introduction to Ehrenfeucht-Frasse Games, Bryan W. Roberts

http://www-bcf.usc.edu/~bwrobert/research/RobertsB_EFGames.pdf
[4] Unity3D Homepage : www.unity3d.com

CONTRIBUTIONS
Swetha Rajagopal : Project ideas, Implementation, Graphics
Mangala Gowri Krishnamoorthy : Project ideas, Implementation