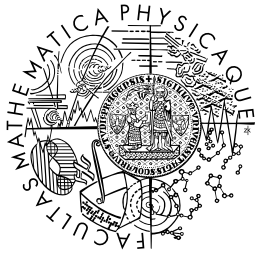

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Daniel Crha

Board game with artificial intelligence

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis: Mgr. Martin Pilát, Ph.D.
Study programme: Computer Science
Study branch: IOI

Prague 2020

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Most of all I want to thank my supervisor for his help and advice, he was always there for me whenever I needed his opinion. I also thank all of my family and friends for being there for me along the way, my journey has been long and I could not have done it without them.

Title: Board game with artificial intelligence

Author: Daniel Crha

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Martin Pilát, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Multiplayer board games with imperfect information present a difficult challenge for many common game-playing algorithms. Studying their behavior in such games can be difficult, because existing implementations of such games have poor support for artificial intelligence. This thesis aims to implement an imperfect information multiplayer board game in a way that provides a framework for developing and testing artificial intelligences for board games with the aforementioned qualities. Furthermore, this thesis explores the implementation of several algorithms for the game. This aims to showcase the artificial intelligence framework, as well as to analyze the performance of existing algorithms when applied to a board game with elements such as hidden information and multiple players.

Keywords: board game, artificial intelligence

Contents

Introduction

Foreword

In game theory, perfect information two-player games are often studied, and numerous algorithms have been designed with the purpose of playing them. This includes games like Chess and Go, which have had large breakthroughs in recent years [1]. However, real world situations do not always have perfect information, or only two parties involved. We could for example imagine multiple countries, which have only approximate information about the armies of their opponents. In this scenario, it could be useful to have tools to simulate potential enemy troop movements or placements.

Even though algorithms which are able to model imperfect information and multiple players are often useful, they are not studied nearly as often. Designing such an algorithm is not easy, and there are many pitfalls which make conventional game theory algorithms much less effecive at solving imperfect information and multi-player problems. This thesis therefore aims to analyze the problems of implementing such algorithms, and to implement some of them in pursuit of that goal.

Naturally, some frameworks do already exist for the implementation of such games. However, at the time of writing, some of them only have AI (Artificial Intelligence) support as an experimental and sparsely documented feature [2], and others only focus on specific fields of AI [3]. This work aims to provide a kind of plug-and-play" experience, where AI developers have minimal barriers between cloning a git repository and having a working AI.

Goals

The main goal of this thesis is to create a multi-player board game with imperfect information states. The game will primarily be designed with AI in mind, and it will provide a reasonable interface for the implementation of AI players.

Another goal is the implementation of several AI players for said game. This will allow us to not only explore potential problems with implementing AIs for games of this kind. We will also verify that the API (Aplication Programming Interface) provided by the game is sufficient for implementation of such AI players, and that the API is reasonably easy to use.

1. Related Work

1.1 Games and Frameworks

1.2 Algorithms

1.2.1 Minimax

1.2.2 Monte Carlo Methods

2. Game Design

3. Framework

3.1 Design

3.2 Interface

4. Used Algorithms

4.1 Random Decisions

4.2 Heuristics

4.3 MaxN

4.4 Information Set Monte Carlo Tree Search

5. Experiment Description

There are two qualities which we want to analyze with respect to the game and the implemented AI algorithms:

- Identify potential asymmetries in game balance
- Compare methodologies used by the AI algorithms

To this end, we conducted five experiments, split according to their purpose. The following sections elaborate on the experiments and their results.

5.1 Game Balance Experiments

As mentioned in chapter 2, the game features a degree of asymmetry. The order in which players take their turns inherently changes the viability of certain strategies, because players in different positions have different information sections available to them. For example, the player in the first position has perfect information about which colonist was removed from play during the colonist pick phase, while the second and third players do not have such certainty.

Most importantly however, a player's colonist is revealed at the start of their turn. This means that if the player in the fourth position is a Spy or an Opportunist, they will know all the other players' colonists when their turn comes around. This means that this player will be able to target any player with their targeted ability without the fear of missing or hitting an unintended target.

With these things in mind, we can hypothesize that players in the earlier positions have an easier time achieving synergy-based strategies, since they get priority when picking colonists. On the other hand, we can also hypothesize that players in later positions will benefit from play based around using targeted colonist abilities.

In Chess, it is widely agreed that the white player has an advantage [4]. Similarly, we aim to discover whether player ordering confers a measurable advantage to any player in *Colonizers*. We will conduct this experiment with the null hypothesis — we assume that there is no significant advantage for any player ordering.

5.1.1 Description

We conducted two experiments in this section. In both of them, four identical AIs played 1000 games against each other. In the first experiment, the AI in question was `RandomIntelligence`, and in the second experiment it was `HeuristicIntelligence`.

All random events were seeded, and the results of the games were captured in JSON (JavaScript Object Notation) files. The results were then parsed and analyzed. The JSON result files can be found in the attached source code, refer to subsection A.2.6 for more information on their location and semantics.

The random seeds used by application components during the experiment were as follows:

- `RandomIntelligence`: seed 42
- `HeuristicIntelligence`: seed 97
- `GameConstants`: seed was changed every game to prevent the same game from being played 1000 times. The seeds were generated by a C# random number generator seeded with 42.

5.1.2 Findings

TODO

5.2 Algorithm Comparison Experiments

We have implemented four algorithms in this thesis — `RandomIntelligence`, `HeuristicIntelligence`, `MaxnIntelligence` and `ISMCTSIntelligence`. In order to determine the qualities of said algorithms, we will analyze their differences, along with their advantages and disadvantages. We will also look at how the algorithms perform in play against each other, with the hopes of determining which algorithm is the most suitable for a game like *Colonizers*.

To start with, we would not expect `RandomIntelligence` to perform well in any kind of mutual play. It is present simply as a benchmark for the performance of other AIs.

The more important benchmark is `HeuristicIntelligence`, since it represents rules which were created by observing humans play the game. Therefore we consider this AI to be a minimum benchmark for other AIs to be competent.

`MaxnIntelligence` is based on the MaxN algorithm [5], which is itself based on Minimax [6]. This AI was adapted for imperfect information games, and it spends a non-trivial amount of computing power on simply exploring possible determinizations of the current game state. If we take that into consideration, along with the fact that the branching factor for *Colonizers* is non-trivial, we would expect `MaxnIntelligence` to perform relatively poorly. The depth of the search trees used could not be reasonably increased beyond 7, due to performance concerns. We expect that any kind of long-term strategy could not be achieved by it since it lacks the necessary exploration depth. The Minimax family of algorithms does however offer very solid insight into the few turns it examines,

therefore we hypothesize that this AI will be primarily good at tactics-based play. The performance of `MaxnIntelligence` against `HeuristicIntelligence` is uncertain, therefore we will follow the null hypothesis and assume that their performances are statistically similar. We also hypothesize that since this AI has a strong foundation for tactical prowess, it should win more often when in later positions (namely third and fourth).

The final AI tested is `ISMCTSIntelligence`. This AI is well-adapted to imperfect information and multiple player environments. Therefore we would expect it to outperform the three aforementioned AIs in most situations. We expect it to play well in most circumstances, regardless of player permutation.

5.2.1 Description

We conducted three experiments in this section. In the first experiment, one of each implemented intelligence played 50 games against each other. This experiment is meant to assess the general playing ability of the AIs. In the second experiment, we let two instances of `HeuristicIntelligence` and two instances of `MaxnIntelligence` play against each other for 50 games. Lastly in the third experiment, we performed the same thing as in the second experiment, but we replaced `MaxnIntelligence` instances with `ISMCTSIntelligence` instances. These two experiments are meant to benchmark the adapted algorithms against the heuristic solution.

All random events were seeded, and the results of the games were captured in JSON files. The results were then parsed and analyzed. The JSON result files can be found in the attached source code, refer to subsection A.2.6 for more information on their location and semantics.

The random seeds used by application components during the experiment were as follows:

- `RandomIntelligence`: seed 42
- `HeuristicIntelligence`: seed 97
- `MaxnIntelligence`: seed 99
- `ISMCTSIntelligence`: seed 15
- `GameConstants`: seed was changed every game to prevent the same game from being played 1000 times. The seeds were generated by a C# random number generator seeded with 42.

5.2.2 Findings

WORK IN PROGRESS

Conclusion

Bibliography

[1] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[2] boardgame.io [online]. `https://boardgame.io`. Accessed: 2020-05-31.

[3] Openai gym [online]. `https://gym.openai.com`. Accessed: 2020-05-31.

[4] W. F. Streeter. Is the first move an advantage? *Chess Review*, page 16, 05 1946.

[5] C. A. Luckhardt and K. B. Irani. An algorithmic solution of n-person games. In *AAAI*, 1986.

[6] I. Millington and J. Funge. *Artificial Intelligence for Games*. Second Edition. Morgan Kaufmann, 2009.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 User Documentation

A.1.1 Installation

A.1.2 User Interface

A.2 Developer Documentation

A.2.1 Prerequisites

A.2.2 Project Structure

A.2.3 Game Engine

A.2.4 Artificial Intelligence

A.2.5 User Interface

A.2.6 Experiments