Poker with Evolving Automata

Leo Muller

2140043

Specification and Planning Document

Swansea University
Prifysgol Abertawe

Department of Computer Science

Adran Gyfrifidureg

30th October 2023

Declaration

Statement 1

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... Leo Muller (2140043)

Date .....

Statement 2

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by citations giving explicit references. A bibliography is appended.

Signed ..... Leo Muller (2140043)

Date .....

Statement 3

The University's ethical procedures have been followed and, where appropriate, ethical approval has been granted.

Signed ..... Leo Muller (2140043)

Date .....

Contents

List of Figures

1 Introduction

In this project, I will explore the outcomes of running simulations on populations of evolving automata, which will play repeated games of Kuhn poker against each other. Depending on their success in these games, they'll procreate, mutate or die out. This project will look at the possible emergence of an evolutionarily stable strategy that may be adopted by the population.

For the purposes of this project the population will be separated in half, a group that will always be player one and another group that will always be player two.

1.1 Motivation

In a world that is increasingly defined by complex inter-dependencies and strategic interactions, the ability to understand and predict the emergence of evolutionarily stable strategies is of paramount importance. The dynamics of strategic decision-making play a pivotal role in determining the success or failure of individuals and populations, from ecological systems to economic markets. This project is motivated by the desire investigate strategy evolution through the lens of evolving automata and game theory. The concept of evolutionarily stable strategies is central to understanding the stability of strategic behaviors within evolving populations. They represent strategies that, once established, resist invasion by emerging strategies, rendering them evolutionary stable. This project is motivated by a desire to potentially derive real-world applications from and understanding the dynamics of strategy evolution.

1.2 Aims and objectives

The primary aim of this project is to explore the outcomes of running simulations on populations of evolving automata engaged in repeated games of Kuhn Poker. The project endeavors to understand how individuals in the population will adapt, evolve, and develop strategies in competitive environments, ultimately seeking to ascertain whether an evolutionarily stable strategy may emerge within the automata population.

The main objectives of this project are:

1. *Simulation framework development*: Develop a robust simulation framework capable of facilitating populations of evolving automata in repeated games of Kuhn Poker. Implement a realistic and scalable environment for automata interaction and strategy evolution.
2. *Automata performance evaluation*: Establish criteria for evaluating the performance of the automata in Kuhn poker games and assess the success of the automata's strategies based on these criteria.
3. *Emergence of evolutionarily stable strategies*: Examine the games played by automata to identify recurring behaviours that exhibit an evolutionary stable strategy within the population. Investigate whether the population converges toward an evolutionarily stable strategy or if multiple strategies coexist.
4. *Comparison with game theory*: Compare the strategies and outcomes observed in the simulations with established game-theoretic concepts such as the Nash equilibrium strategy.

Determine the extent to which automata strategies align with or diverge from these theoretical benchmarks.

5. *Practical applications and implications*: Explore the potential practical applications of findings, including relevance to artificial intelligence, autonomous systems, and decision-making processes. Discuss the implications of the research for understanding complex adaptive systems in computational modeling.

1.3 An overview of Kuhn poker

Poker is zero-sum imperfect-information game which can be played by two or more participants. It also incorporates a move by nature in the form of a dealer, this is interesting as it means the automata must adapt to handle random possibilities.

Kuhn poker is a simplified variant of poker developed by Harold W. Kuhn as a simple model two-player game, amenable to a complete game-theoretic analysis [1].

The list of changes that facilitate these features of Kuhn poker are as follows:

- The deck for Kuhn poker only consists of three playing cards, a King, Queen, and Jack.
- The players can only bet in increments of one unit;
- The ability of the player to raise has been removed.

In order to limit the complexity of the automata, the interactions for Kuhn poker will also be changed. Instead of being able to *bet*, *check*, *call*, and *fold*, the automata will have the choices of **bet** and **pass**. Here **bet** functions as both *bet* and *call*, and **pass** functions as both *check* and *fold*. Therefore, the second **bet** made by any player will function as a *call*, and any **pass** made in response to a **bet** functions as a *fold*.

Figure 1 shows a tree of all moves that the automata can make in a game Kuhn poker.

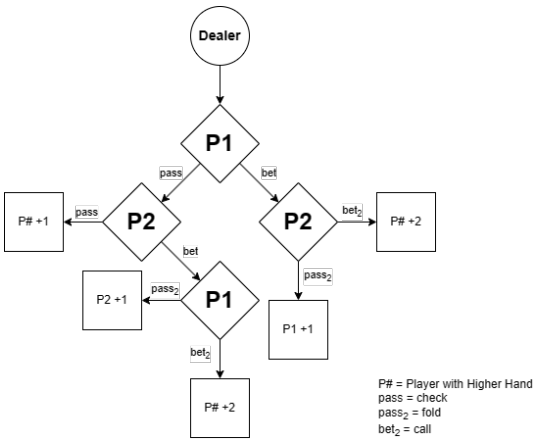

Figure 1: Tree for game of Kuhn poker

1.4 Evolutionary programming

Evolutionary programming is an evolutionary algorithm paradigm first used by Lawrence J. Fogel in 1962. Fogel used finite state machines to solve a prediction problem. These machines would be mutated by adding or deleting states, or changing the state transition rules. The best of these mutated machines would be evolved further in future generations. After being successfully applied to prediction problems, system identification, and automatic control, the evolutionary programming method was extended to handle time series data and to model the evolution of gaming strategies[2].

2 Solution concepts

A solution concept is a formal rule used for making game-play predictions known as ”solutions”. Solutions describe which strategies the players will adopted by players and, therefore, predict the outcome of the game. Equilibrium concepts are the most commonly used solution concepts, Nash equilibrium being the most famous.

2.1 Nash equilibrium

In game theory, the Nash equilibrium is the traditional solution concept for non-cooperative games involving multiple players. In a Nash equilibrium, each player is assumed to have common knowledge of the equilibrium strategies of the other players, and there does not exist a player that would gain anything by changing their own strategy[4].

Given a two player game, let *E(S,T)* represent the payoff for playing strategy *S* against strategy *T*. The strategy is a Nash equilibrium only if for both players and for any strategy *T*:

$E(S,S) \geq E(T,S)$

Using this definition, a strategy $T \neq S$ can score equally well, but not better to *S*. In this case, a Nash is presumed to be stable. This is presumed on the assumption that there is no incentive for players to adopt *T* instead of *S*.

2.2 Evolutionarily stable strategy

An evolutionarily stable strategy is an equilibrium refinement of the Nash equilibrium. Thus, once established by a population, natural selection will prevent any strategies that emerge from mutation from replacing it.

The conditions for a strategy $S \neq T$ to be an evolutionarily stable strategy are:

1. $E(S,S) \geq E(T,S)$, and
2. $E(S,T) > E(T,T)$

The first condition states that the strategy is a Nash equilibrium and the second condition establishes that the population of players who play strategy S has an advantage when playing against T[7].

3 Related work

My project explores the possible emergence of an evolutionarily stable strategy in games of Kuhn poker when played by populations of evolving automata. When doing my research I found two similar works that I felt worth mentioning.

3.1 Evolving Nash-optimal poker strategies using evolutionary computation

The first is a thesis on simplified poker and the evolutionarily stable strategy[5]. It focuses on using evolutionary algorithms to development competitive computer players that specialize in heads-up Texas Hold'em poker. In their thesis, Quek design an evolutionary model for the purpose of achieving strategies that play at Nash equilibrium. Quek successfully used an evolutionary algorithm to generate a player that displayed logical strategies and also show insights into bluffing indicators.

3.2 Evolving automata using genetic programming

The second is a thesis on evolving automata using genetic programming[3]. This thesis investigates genetic programming as a method to automatically evolve various classes of automata. Unlike my project which touches on game theory, this paper solely focuses on evolving automata. Additionally, it cavers a much broader range of automata including finite acceptors, push-down automata, finite state transducers and Turing machines. When implementing her automata Amashini Naidoo also chose to use Java as the language. It isn't explained in her thesis why she implemented it on Java, but it is likely due to Java being an object-orientated language with similar syntax to C and C++.

The aim of Naidoo's thesis was to determine if genetic programming is an effective method for inducing automata. genetic programming did prove to be a highly effective method of inducing automata, therefore, if evolutionary programming proves to be too limiting a method for inducing evolving automata, I will consider using genetic algorithms instead.

4 Project management

This project requires that I implement an evolutionary algorithm as an evolving finite automata, and develop a functional implementation of this algorithm in java.

This automata is be defined as follows:

- P = { Q, Σ, q, δ }
- Q = {Deal, Ante, Bet, Pass}
- Σ = {king, queen, jack, ante, bet, pass}
- q = {Deal}
- δ = Q x Σ –> Q

An example of an automata following this design would be Figure 2 which shows a finite automata for the equilibrium strategy of the second player.

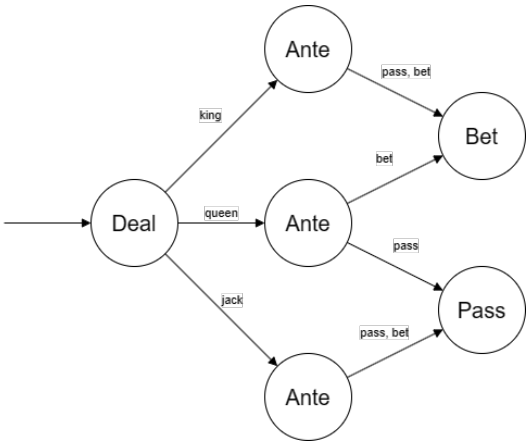

Figure 2: Finite automata for Player 2 following equilibrium strategy

Additionally, I also require a program capable of facilitating populations of evolving automata in repeated games of Kuhn Poker. For ease of interaction between the automata and the environment, this will also be developed in java.

4.1 Project methodology and scheduling

I've opted for an Iterative Development model for this project. Unlike traditional approaches that demand a comprehensive specification plan upfront, this model is designed to start with minimal initial requirements, focusing on both specification and implementation of only a portion of the software. During development, the prototype is subject to continuous review, allowing the incorporation of additional requirements. This iterative process then evolves into subsequent versions of the application.

The Iterative development model is comprised of four repeating stages. These are:

1. *Requirements Phase*: Project scope is identified, the requirements and risks for the current iteration are can be estimated.
2. *Design Phase*: A functional design that fulfills the requirements is established. This may be an update/extension of the previous design.
3. *Implementation and Test*: The system is developed and updated to incorporate any new features. The entire system the undergoes testing.
4. *Review Phase*: The software is reviewed in accordance with the requirements. Then, further requirements are reviewed and proposed for an update in the next iteration.

At the end of every cycle the system is reviewed to establish whether it will be accepted as the deliverable system, or rejected and undergo further iterations[6].

Figure 3 shows the work schedule using a Gantt chart.

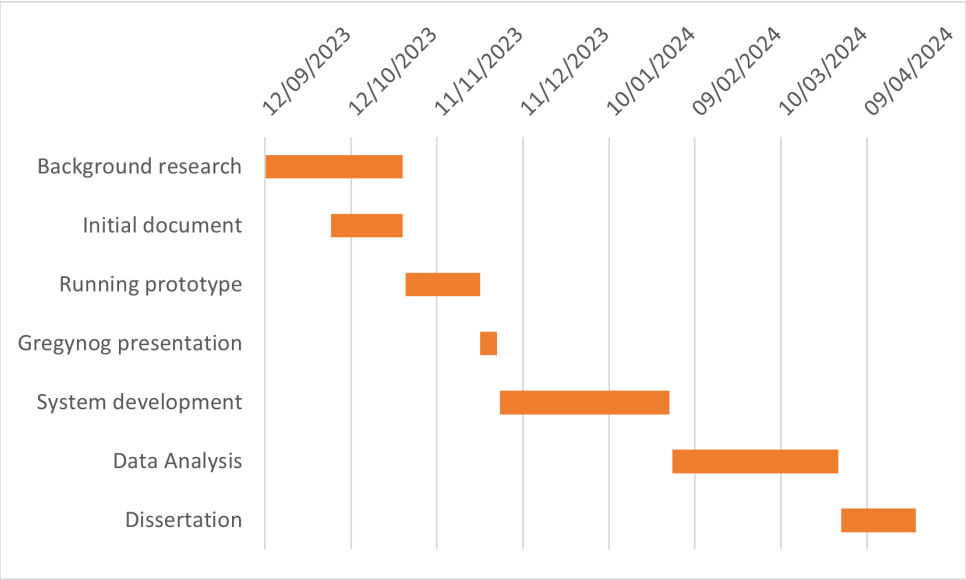

Figure 3: Work schedule for dissertation

4.2 Project risk analysis

It may be found that evolutionarily stable strategies won't emerge from Kuhn poker. While this wouldn't prevent the automata from displaying a variety of other successful strategies, this would limit our ability to analyse the automata though the lens of game theory. To solve this issue, the automata in this project could be updated to play a more complex version of poker. If the automata was designed to accept poker hands as an alphabet instead of combinations of poker cards, it would lower the alphabet to a length of ten rather than the total number of all possible poker card combinations. Then, if an interface system was put in place to analyze what poker hand an automata has and could return that information, the automata would only have to be able to handle ten different inputs rather than having to process all the different poker card combinations.

Additionally, finite automata may not capable enough to run games of poker. This would be detrimental to the project as it would prevent the automata for being able to effectively play poker. To combat this, pushdown automata could be used instead of finite automata. A pushdown automata could accept an alphabet consisting of moves from the opponent and a stack of moves made by the dealer. The automata could be initialised with a stack containing the hand dealt to it.

Figure 4 shows figure 2 translated into a pushdown automata.

Ignoring the limitations of the capability of finite automata, using pushdown automata would greatly reduce the number of states required to run the same algorithm. This would also allow for the automata to handle more complex forms of poker, possibly even a full variant if the automata accepted poker hands as part the stack rather than cards. Using a more complex poker variant would, however, make it more difficult to analyze what strategies the automata are using as full poker has a much larger variation of optimal strategies.

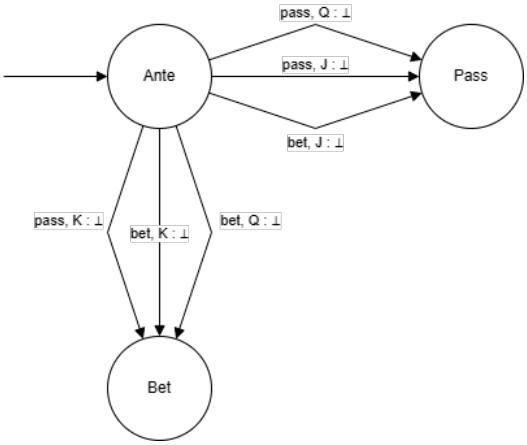

Figure 4: Pushdown automata for Player 2 following equilibrium strategy

It is a possibility that evolutionary programming is too limiting a paradigm to develop evolving automata for the purpose of playing poker. This would prevent the automata from effectively completing the tasks required of them. This is easily amended though, as there are a plethora of alternative evolutionary algorithms that can be used. For example, genetic programming could be used, which would add an extra layer of mutation depth as they reproduce - merge parts of two other programs to create a child program - as well as mutate.

References

[1] H. F. Bohnenblust et al. *Contributions to the Theory of Games (AM-24), Volume I*. Red. by H. W. Kuhn and A. W. Tucker. Princeton University Press, 1952. ISBN: 978-0-691-07934-9. URL: https://www.jstor.org/stable/j.ctt1b9rzq1 (visited on 28/10/2023).

[2] David B. Fogel. *Evolutionary computation: the fossil record*. OCLC: 38270557. New York: IEEE Press, 1998. 641 pp. ISBN: 978-0-7803-3481-6. URL: http://catdir.loc.gov/catdir/toc/onix07/98002728.html (visited on 29/10/2023).

[3] Amashini Naidoo. 'Evolving Automata Using Genetic Programming'. In: ().

[4] Martin J. Osborne and Ariel Rubinstein. *A course in game theory*. Cambridge, Mass: MIT Press, 1994. 352 pp. ISBN: 978-0-262-15041-5 978-0-262-65040-3.

[5] Hanyang Quek et al. 'Evolving Nash-optimal poker strategies using evolutionary computation'. In: *Frontiers of Computer Science in China* 3.1 (1st Mar. 2009), pp. 73–91. ISSN: 1673-7466. DOI: 10.1007/s11704-009-0007-5. URL: https://doi.org/10.1007/s11704-009-0007-5 (visited on 29/10/2023).

[6] *SDLC Iterative Model*. URL: https://www.w3schools.in/sdlc/iterative-model (visited on 29/10/2023).

[7] Bernhard Thomas. 'On evolutionarily stable sets'. In: *Journal of Mathematical Biology* 22.1 (1st June 1985), pp. 105–115. ISSN: 1432-1416. DOI: 10.1007/BF00276549. URL: https://doi.org/10.1007/BF00276549 (visited on 30/10/2023).