INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA INFORMÁTICA

Aprendizagem máquina aplicada ao contexto do Poker

TIAGO SILVA MARTINS

Outubro de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Machine learning applied to the context of Poker

Tiago Silva Martins

Dissertation to obtain the Master Degree in Informatics Engineering, in the Graphical Systems and Multimedia area of expertise

Advisor: Carlos Vaz de Carvalho

Jury:

President:

Vocal:

Porto, October 2020

Resumo

A combinação de princípios da teoria de jogo e metodologias de *machine learning* aplicados ao contexto de formular estratégias ótimas para jogos está a angariar interesse por parte de uma porção crescentemente significativa da comunidade científica, tornando-se o jogo do Poker num candidato de estudo popular devido à sua natureza de informação imperfeita.

Avanços nesta área possuem vastas aplicações em cenários do mundo real, e a área de investigação de inteligência artificial demonstra que o interesse relativo a este objeto de estudo está longe de desaparecer, com investigadores do *Facebook* e *Carnegie Mellon* a apresentar, em 2019, o primeiro agente de jogo autónomo de Poker provado como ganhador num cenário com múltiplos jogadores, uma conquista relativamente à anterior especificação do estado da arte, que fora desenvolvida para jogos de apenas 2 jogadores.

Este estudo pretende explorar as características de jogos estocásticos de informação imperfeita, recolhendo informação acerca dos avanços nas metodologias disponibilizados por parte de investigadores de forma a desenvolver um agente autónomo de jogo que se pretende inserir na classificação de *"utility-maximizing decision-maker".*

Keywords: Machine Learning; Game Theory; Poker; Strategy Games; Probabilities; Opponent Modeling;

Abstract

The combination of game theory principles and machine learning methodologies applied to encountering optimal strategies for games is garnering interest from an increasing large portion of the scientific community, with the game of Poker being a popular study subject due to its imperfect information nature.

Advancements in this area have a wide array of applications in real-world scenarios, and the field of artificial intelligent studies show that the interest regarding this object of study is yet to fade, with researchers from Facebook and Carnegie Mellon presenting, in 2019, the world's first autonomous Poker playing agent that is proven to be profitable while confronting multiple players at a time, an achievement in relation to the previous state of the art specification, which was developed for two player games only.

This study intends to explore the characteristics of stochastic games of imperfect information, gathering information regarding the advancements in methodologies made available by researchers in order to ultimately develop an autonomous agent intended to adhere to the classification of a utility-maximizing decision-maker.

Keywords: Machine Learning; Game Theory; Poker; Strategy Games; Probabilities; Opponent Modeling;

Acknowledgements

I offer my utmost gratitude to my advisor, Doctor Carlos Vaz de Carvalho, for allowing, supervising and providing this study with his expertise, as well as all the teachers in *Instituto Superior de Engenharia do Porto* and *Universidade de Aveiro* who shaped me as a professional and as a person.

To my family, friends and colleagues I have nothing but gratitude for the support, motivation and guidance I have been provided with throughout my academic journey and the development of this study.

Finally, I owe my gratitude to my employers, who motivate my critical analysis and scientific curiosity on a daily basis.

To everyone who has supported my academic path in any way, I pay my best regards.

Table of Contents

List of Figures

List of Tables

Acronyms and Glossary

Acronyms List

AF	Aggression factor (2.2.8)
AHP	Analytical Hierarchy Process (4.1.4)
AI	Artificial intelligence
BB	Big-blind (2.2.1)
BTN	Button (2.2.1)
CI	Consistency index (4.1.4)
CO	Cutoff (2.2.1)
CR	Consistency ratio (4.1.4)
EGT	Evolutionary game theory (2.4.2)
FL	Fixed Limit (2.3)
HJ	Hijack (2.2.1)
IP	In-position (2.2.1)
ML	Machine learning
MP	Middle Position (2.2.1)
NCD	New concept development (0)
NPD	New product development (0)
NE	Nash equilibrium (2.1.4)
NL	No Limit (2.3)
OOP	Out-of-position (2.2.1)
QFD	Quality Function Deployment
VPIP	Voluntarily put $ in the pot (2.2.8)
PFR	Preflop raise (2.2.8)
PL	Pot Limit (2.3)

RFI	Raise first in (2.2.8)
RFE	Required fold equity (2.2.7)
RI	Random index (4.1.4)
SB	Small-blind (2.2.1)
VA	Value analysis (0)
VE	Value engineering (0)

Symbols list

$\boldsymbol{i}$	Player
$-\boldsymbol{i}$	Opponent
$\boldsymbol{N}$	A set of n players
$\boldsymbol{A}$	A single set of actions
$\boldsymbol{H}$	A set of non-terminal decision nodes
$\boldsymbol{Z}$	A set of terminal decision nodes, disjoint from $\boldsymbol{H}$
$\boldsymbol{s}$	Mixed strategy profile
$\boldsymbol{s}^*$	Mixed strategy profile in Nash equilibrium
$\boldsymbol{F}$	A player's strategy set
$\boldsymbol{\chi}$	Action function
$\boldsymbol{\rho}$	Player function
$\boldsymbol{\sigma}$	Successor function
$\boldsymbol{\mu}$	Utility function

1 Introduction

This chapter introduces the project, providing a high-level description, contextualizing it and describing its main objectives. The problem statement and solution hypothesis are presented, as well as the procedures that will be adopted throughout the design and development process of the solution. Finally, it describes the structure of this document.

1.1 Context

In computer science, artificial intelligence (AI) is used to define the field of study of intelligent agents – any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals (Poole, Mackworth and Goebel, 1998). The application of game theory concepts in association with machine learning algorithms constituting an interesting and widely publicized field of scientific investigation, with the knowledge acquired through the development of algorithms in academic context finding ample application in real-world scenarios.

The importance of games is demonstrated by the fact that strategic games were one of the first sub-domains to be studied in artificial intelligence, initially with the sole objective of finding winning strategies, and posteriorly with the goal of maximizing agents' utility, a term which allows for a deep definition of certain game assets and characteristics, and one which has been a target of a significant amount of studies.

Games such as Checkers and Chess have been a publicized object of game theory studies for decades, with renowned scientist formulating extensive theories regarding them and developing intelligent game playing agents. The set of rules and goals that characterize these games render them as advantageous test subjects for artificial intelligence approaches, enabling the accurate measurement of the agent's success and thereby the solution's validation.

The aforementioned games, however, are in many ways unrelated to real-world events due to the completeness of the information available and, on the contrary, the lack of stochasticity, characteristics which real-world decision-makers face daily when acting in competitive environments.

Poker and other games deemed of incomplete information provide a domain for artificial intelligence studies in which the knowledge obtained, and methodologies developed are passible of transferring to scenarios separate from the game's universe. This, along with other characteristics associated with the game of Poker such as the opponent modelling, calculated risk management and its recent media coverage, makes it a common domain for artificial intelligence studies.

1.2 Problem

The game of Poker is considered a major challenge in the field of artificial intelligence due to factors in the nature of its domain, the main of which are the inherent imperfect-information and the size of the game tree.

Recently, Facebook and the Carnegie Mellon University's joint effort produced Pluribus (Brown, 2019), the first Poker-playing agent proved capable of defeating human players in a setting where the game isn't limited to 2 players, with all previous studies culminating in solutions aimed at a heads-up game in which the agent is playing against a single human.

Despite the developments presented in recent years, few takeaways were able to be extracted from the research for human players to implement in their game, with factors such as the following making it extremely difficult to construe strategic insights that can be generalized:

- Incomplete information;
- Size of the decision tree (depending on the game's betting structure);
- Hand evaluation, from which only an estimate can be obtained;
- Low number of observations;
- Partial results;
- Multiple equilibrium's (on a multiplayer setting).

1.3 Hypothesis

When analyzing a problem, it becomes important to formulate a hypothesis for a solution, so that this hypothesis can be posteriorly tested in order to conclude about the project achievement level.

The hypothesis proposed by this study is that machine learning and game theory methodologies can be applied in order to develop a game playing agent that can compete against human players in the game of Poker. This agent should adhere to specified standards of computational and in-game performance, functioning in the form of an API that is capable of outputting a response action to a given in-game situation that is fed to it via HTTP or other communication protocol, and store a record of its games and actions for posterior evolutionary analysis of results and tendencies.

1.4 Goals

The main goal of this dissertation is to study different machine learning approaches based on probability, rules and events models, as well as their conjunction with game theory methodologies of situational probabilistic calculation, estimative of utility, opponent modeling and risk management in order to formulate decisions in response to in-game situations.

This goal is accompanied by the secondary objectives of developing a HTTP-REST service that implements a selection of the previously mentioned methodologies for the accomplishment of the main goal, which is to be coupled with a web-interface through which a human is able to play against the solution’s agent.

1.5 Procedures

1.5.1 Methodology

A methodology based on the principles of agile development will be used in the development of the solution, focusing on features, user stories and test cases as instructive forms of documentation and guidance.

1.5.2 Version control

A software development version control system will be used in order to manage and safeguard the solution’s source code along its development lifecycle, namely Git, through

GitHub’s infrastructure. GitHub is a distributed version control and source code management platform that bundles basic Git functionality with proprietary features such as task management, bug tracking and feature requests among other functionality in an integrated and easily accessible package.

1.5.3 Task management and planification

GitHub’s integrated features will be used in order to assess tasks’ and overall project completion, report problems in specified components and compile the solution’s documentation artifacts.

In accordance to agile principles, the development lifecycle will be planned in an iterative and incremental manner, with a defined duration for sprints, each of which will be attributed a number of tasks to complete. The development expectations for each sprint shall be adjusted along iterations in order to improve the measurement of expectations and improve the accuracy of the measure in which designated tasks can be accomplished within a sprint’s timeframe.

1.6 Document Structure

This document contains the following sections, which can be divided into 3 separate groups:

The Theoretical knowledge and State of the Art chapters present the theoretical domain on which this study focuses; The Value analysis, Design and Development chapters define the methodology and artifacts based on which a solution for the presented hypothesis is to be implemented and discuss said implementation; Experimentation and evaluation and Conclusion chapters evaluate the solution and discusses the obtained results.

The definition of each chapter’s content is as follows:

- **Theoretical knowledge** presents the fundamental theoretical concepts and background material considered to be indispensable for the understanding of this study. This material relates to game theory in general and its application to the game of Poker;

- **State of the Art** contextualizes solutions that share similarities with the one this study aims to develop and describes general applications that are relevant to this study;

- **Analysis** presents an analysis of the project’s value through the identification and analysis of use cases, scenarios and requirements. Beyond this, it defines metrics to evaluate the fulfillment of requirements and justifies the decisions made in regard to the selection of technologies;

- **Design** describes the design of the business and data models, as well as the architectural design of the product, based on current standards and good practices established for web and software development;
- **Development** reports the challenges faced and decisions made during the development process of the solution;
- **Experimentation and evaluation** introduces the testing methodology and parameters based on which the evaluation of the solution will be conducted, and finalizes by analyzing the results of the evaluation;
- **Conclusion** provides an overview of the project’s achievements and limitations, as well as discussing some ideas for further development.

2 Theoretical knowledge

In order to understand the concepts involved in this project, it is vital to acquire theoretical knowledge regarding the business concepts and their formalization. Concepts presented become less generic as this chapter progresses, relating specifically to the object in study at it's ending.

Terms specific to the game of Poker are available for consultation in Appendix A – Poker glossary.

2.1 Game theory

Game theory is the study of strategic interaction through the use of mathematical models which translate the conflict and cooperation transactions among rational decision-makers in an environment with defined characteristics (Myerson, 1991). This field has a wide range of applications, from economical to political and social, aiding agents in the analysis of complex decisions that involve trade-offs between the individuals involved in a system. Quoting Roger B. Myerson, the situations that game theorists study are not merely recreational as the term might suggest, with "Conflict analysis" or "interactive decision theory" being more accurate terms. Modern game theory was initially introduced by John von Neumann, proving the concept of mixed-strategy equilibria in two-person zero-sum games in 1928 (John von Neumann, 1944).

This field of study generally aims to understand conflict and cooperation by studying quantitative models and hypothetical examples which can be applied of in order to predict or produce proactive results.

Some degree of knowledge regarding the linguistics and assumptions associated with the field of game theory is necessary in order to correctly interpret this study. Defining general terminology according to (Myerson, 1991), the term *game* refers to any social situation involving two or more individuals, who are termed *players*. Players are assumed to be rational and intelligent, adjectives which carry a technical meaning according to game theorists.

A player is deemed rational if he makes decisions that consistently aid the achievement of his objective, which is to maximize the expected value of his payoff, measured in a defined scale of *utility*. The assumption based on which the adjective of intelligence is attributed to the player is that he has full knowledge regarding the game and is capable of making every inference possible given the available information.

A decision-maker's *expected utility* is defined via its preferences through mathematical methods in which quantitative model is used to describe his behavior. Decision theory demonstrates that in order to satisfy certain intuitive axioms, rational decision-makers should behave – have preferences – in a way that maximizes the expected value of a utility function.

"(…) Any rational decision-maker's behavior should be describable by a utility function, which gives a quantitative characterization of his preferences for outcomes or prizes, and a subjective probability distribution, which characterizes his beliefs about all relevant unknown factors"

(Myerson, 1991)

Decisions applied to events that involve chance are commonly described by probability models or state-variable models, both of which pertain a situation where a decision-maker is choosing among lotteries, varying on how the lottery is defined. In probability models, a lottery is a probability distribution over a set of prizes, while in state-variable models it is a function from a set of possible states into a set of prizes. A game of roulette is dependent on events which are characterized as objective unknowns due to having obvious objective probabilities, while a wager on the result of a sports event is dependent on events which are characterized as being subjective unknowns due to the opposite reason – the latter sees it's prize determined by unpredictable events for which probabilities cannot be specified.

2.1.1 Game classification

Games can be classified according to characteristics intrinsic to its environment, interactions and outcomes, which impact the game's state and decision-makers behavior.

- **Cooperative / Non-cooperative:** A game is cooperative if players are able to use contracts which are specifically modeled in the game and are enforceable by a third party (e.g. judge). Players are still able to cooperate in games classified as non-cooperative, with the basis of differentiation being that in games termed as such, contracts must be self-enforcing (Shor, 2006).

- **Zero-sum / Non-zero sum:** A game is zero-sum if a player's winnings equal other players losses, implying that player's choices cannot increase or diminish the available resources. In games with such classification, the total benefit to all players in the game, for every combination of strategies, always adds to zero. In non-zero-sum games, such as the prisoner's dilemma, the outcome has non-zero net results (Sfetcu, 2014).

- **Discrete / Continuous:** A game is discrete if it has a finite number of elements that compose its environment (players, moves, events, outcomes, etc.). This definition is not to be confused with finitely long games – although a game classified as discrete being also accurately classified as finitely long game, the latter classification defines a game where the winner or payoff is unknown until all the moves are completed.

- **Symmetric / Asymmetric:** A game is symmetric if the payoff obtained by employing a certain strategy would remain unaltered if the players employing it was switched.

- **Deterministic / Stochastic:** A game is deterministic if a set of actions input by the player(s) always outputs the same result, and stochastic if the response to a player's actions can be influenced by chance events, which ultimately affect the winner or payoff.

- **Sequential / Simultaneous:** A game is sequential when players are aware of previous players' actions, and simultaneous when players act at the same time, or the later players are unaware of the previous actions taken.

- **Perfect information / Imperfect information:** A game is of perfect information if every player has knowledge of all the events that previously occurred, including the "initialization event" – player's starting hands in a card game, for example (Muthoo, Osborne and Rubinstein, 1996).

- **Complete information / Incomplete information:** A game is of complete information if every player has knowledge of the strategies and payoffs available to other players – in this case, it is said that the state of the game is globally available.

2.1.2 Representation

The standard form of game representation varies in accordance to the classification assigned to it in terms of being sequential or simultaneous.

Simultaneous games, also termed normal form games or strategic games, are typically denoted in the form of a payoff matrix composed by the strategy set and corresponding payoffs for each player.

Table 1 - Prisoner's dilemma payoff matrix

A \ B	**B stays silent**	**B betrays**
A stays silent	(-1, 1)	(-3, 0)
A betrays	(0, 3)	(-2, 2)

This form of representation, termed normal form representation, is put in use to identify strictly dominated strategies and Nash equilibria.

In turn, sequential or extensive-form games are typically denoted in the form of a decision tree, in which each player's decision points are placed and connected in accordance to every possible outcome his actions lead to. The tree commences at the decision point which will be referred to as *root*, controlled by the player who acts first (Tipton, 2012). This decision point is connected to other nodes through lines, each of which represents an action that this player can possibly perform at this point of the game.

The set of actions taken by the player ultimately leads to an end node in which the play reaches its conclusion and payoffs are defined, which will be referred to as *leaf*. The line connecting the root to the leaf which represents the player's strategy in the given play is often called *line* as per Poker terms (*Line - Poker Terms Glossary | PokerStrategy.com*, 2020), an important definition for this study.

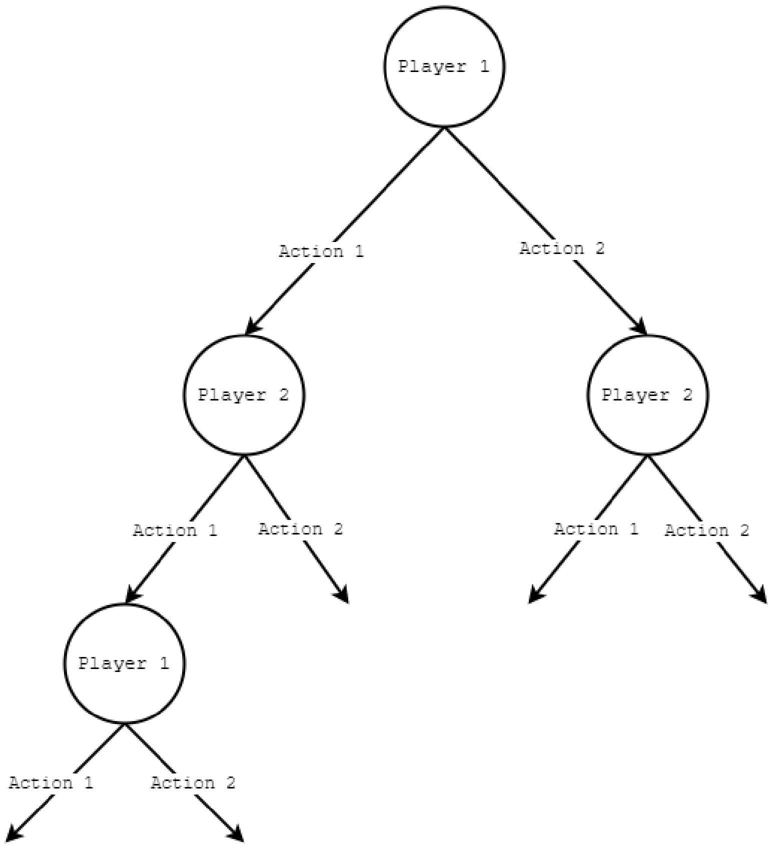


Figure 1 – Decision tree as a form of extensive form game representation

Unlike normal form representation, extensive-form representation does not omit information in the representation, preserving aspects such as the sequence of possible actions and the player's choices along the game.

2.1.3 Strategy

In game theory, a decision-makers behavior is translated by his strategy, which is the options chosen in a setting where the outcome is influenced by all player's options (Polak, 2007). While the concept of move refers to the action chosen by the player at a certain point of a game, the concept of strategy refers to the algorithm through which a decision-maker act throughout every possible situation in the game. Strategy can therefore be described in the form of a function that receives the game state[1] input and outputs an action, which is the player's decision, and is commonly divided among 3 studied types:

- **Pure strategy:** One action is mapped to each possible state. A move is defined for every possible game state, and the player never deviates.

[1] Information set containing all the game's information that is known to the player.

- **Mixed strategy:** Each possible state is mapped to a probability distribution over actions. A probability is assigned to each pure strategy, and the player randomly selects one.

- **Totally mixed strategy:** Each action is mapped to varying states with positive probability.

A player's strategy set is composed by all the pure strategies available to them and is always a subset of all the possible strategies. The combination of every player's strategy sets forms the game's strategy, which defines all the possible paths in a game.

2.1.4 Nash equilibrium

John Forbes Nash, a young mathematics graduate student at Princeton University, formulated the notion of equilibrium that is named after him in a 1950 communication to PNAS[2], in what became known as the Nash equilibrium theory (Nash, 1949).

The Nash equilibrium (NE) is a proposed solution to non-cooperative games involving n players with a set of mixed-strategies, in which it is assumed that every player knows the equilibrium strategy of the remaining, and there is no incentive for any player to deviate from his strategy – in fact, the player doing so will encounter a decrease in his overall utility. Consequently, every Nash equilibrium strategy is the best response strategy[3] to all others in that equilibrium.

Nash equilibrium is formally defined as per the following equation (Van Benthem and Ter Meulen, 2011):

$$\forall i \in N, \forall s_i \in F_i\,, \mu_i\left(s_i^*, \underline{s}^*{}_{-i}\right) \geq \mu_i\left(s_i, \underline{s}^*{}_{-i}\right) \qquad (1)$$

, where:

- i represents a player;
- $-i$ represents an opponent;
- N represents the set of players involved in the play;
- F_i represents the strategy set of player i;

[2] Proceedings of the National Academy of Sciences of the United States of America
[3] The best response strategy is that with the largest utility for a given player, given the equilibrium strategies of the remaining players.

- μ_i represents a function which takes strategy sets as arguments and returns the average utility for player i given both strategies;

- s_i represents a mixed strategy profile for player i;

- s_i^* represents a mixed strategy profile for player i which is in Nash equilibrium;

Nash equilibrium strategies are typically computed via self-play algorithms or linear programming, forming an output of 2 strategies, one for each player involved (Teófilo, 2016). The output strategies assure maximum possible utility to the player employing it, guaranteeing that opponents cannot perform better than a given utility value.

2.1.5 Formalization

Consider game G to be a finite, extensive-form game for which a sequential decision problem can be represented in the form of a decision tree in which the root and leaf nodes represent a decision and the nodes forming the line of play represents the sequence of performed actions. This sequence, termed history, can be denoted by h such that $h \in H$.

Game G can therefore be represented in the format of the following tuple (Jackson, Leyton-Brown, 2013):

$$(N, A, H, Z, \chi, \rho, \sigma, \mu) \tag{2}$$

, where:

- N is a set of n players;

- A is a single set of actions;

- H is the set of non-terminal decision nodes;

- Z is the terminal nodes, disjoint from H;

- χ is the action function which assigns a set of possible actions to each decision node such that $\chi : H \rightarrow 2^A$;

- ρ is the player function which assigns to each non-terminal node h a player $i \in N$ who chooses action at $h \mid \rho : H \rightarrow N$;

- σ is the successor function which maps a choice node and an action to a new choice or terminal node[4] such that for all $h_1, h_2 \in N$ and $\sigma(h_1, a_1) = \sigma(h_2, a_2)$ then $h_1 = h_2$ and $a_1 = a_2$;

- μ is a utility function for player i on the terminal nodes Z, such that $u = (u_1, \dots, u_n); u : Z \to \mathbb{R}$.

The successor function combined with the nodes defines a tree, with nodes encoding history. N, A, H, Z describe of the game's tree and history, while χ, ρ, σ, μ defines actions, succession and utility.

There is a utility function for every player, which assigns a real value to a player on every terminal node, which translates the player's payoff in that node.

2.2 The game of Poker

Poker is a generic name used by most people when referring to the game of Texas Hold'em, the most popular of its many variants. This game gained its popularity amidst the rise of online Poker, an ascend believed to be fueled by its coverage in television shows while being played in major events such as the World Series of Poker. It's moderate learning curve and fast speed of play allows even novice players to quickly pick up the game and play a significant number of hands per hour, which was not the case in the previous most played format, 7 Card Stud.

While there are hundreds of variants available, most of them share characteristics despite being played with a different set of rules. Additionally, there are different betting structure rules available for each individual variant, although these only influence the moments in which players are allowed to post bets, and in which amounts. For the purpose of this study, we will focus on the game of Texas Hold'em.

2.2.1 Rules

This game is played using a deck of 52 cards, leaving both jokers out. The lowest card is the deuce and the highest is the ace, with cards higher or equal than 10[5] being often referred to as "broadway cards", which are ordered in the following order, from left to right:

[4] The successor function combined with the nodes defines a tree. Therefore, for all pairs of decision nodes and for every possible action that can be taken, the only way for the successor function to be equal in both cases is if both choice nodes and actions were the same.

[5] The card 10 is referred to as 'T', maintaining a single-character notation for all the cards in the deck - {2,3,4,5,6,7,8,9,T,J,Q,K,A}.

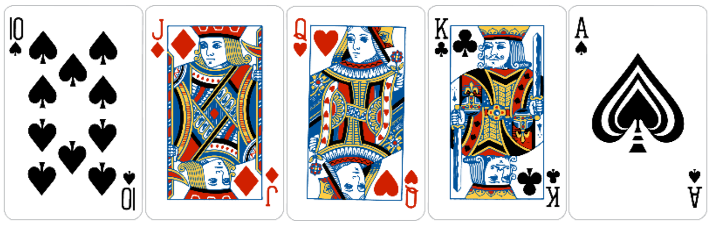


Figure 2 - Broadway cards

Game tables are composed by any number of players ranging from 2 to 10, with the most common formats being:

- **Heads-up –** 2 players
- **6-max –** 6 players
- **9-max**[6] **–** 9 players

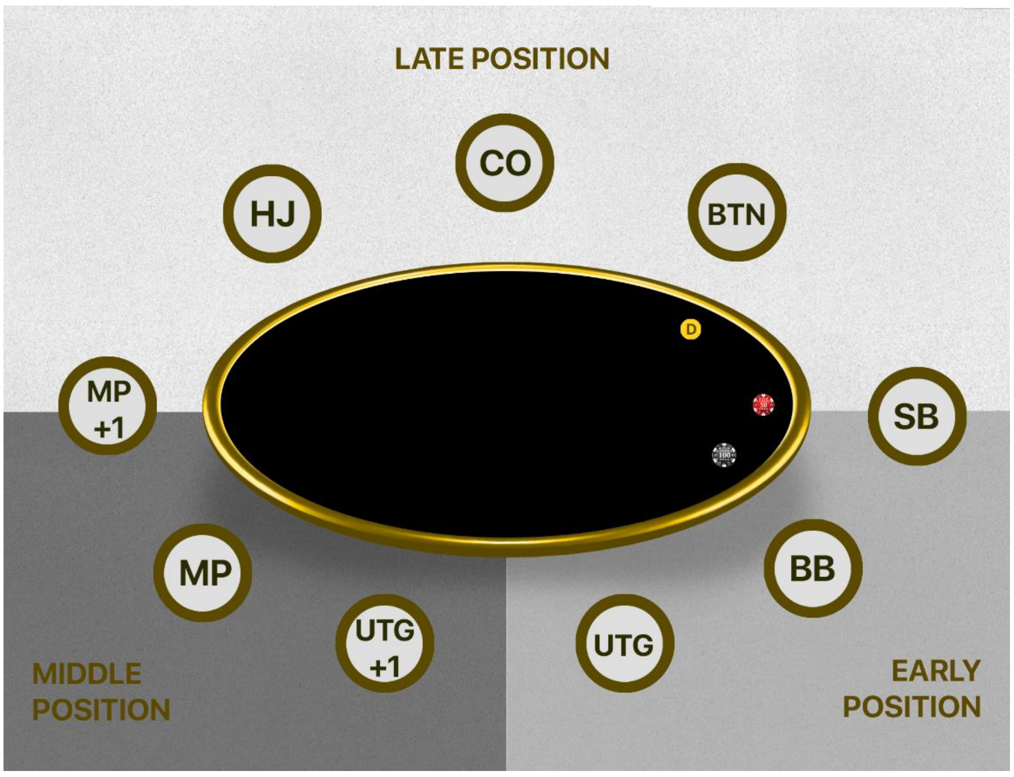


Figure 3 - Table positions

Each table position has a nomenclature, and players move forward one position clockwise at the conclusion of each play.

[6] Tables with a number of players ranging from 7 to 10 are referred to as "Full-Ring".

- **UTG –** Stands for **Under The Gun**. This player is first to act on when a hand is initially dealt. The positions between this and middle position can be referred to UTG+1 and UTG+2;

- **MP –** Stands for **Middle Position**. The positions between this and the Hijack can be referred to as MP+1 and MP+2;

- **HJ –** Stands for **Hijack**. The player in this position is to the left of Cutoff;

- **CO –** Stands for **Cutoff**. The player in this position is to the left of the Button;

- **BTN –** Stands for **Button**, also referred to as **Dealer**;

- **SB –** Stands for **Small-Blind**. This player must place half a minimum bet and is first to act on betting rounds posterior to the first;

- **BB –** Stands for **Big-Blind**. This player must place a minimum bet on the first betting round;

A play in the game of Poker is often referred to as a "hand", and at the beginning of each, the players positioned in the small-blind and big-blind have to post the minimum-bets according to the table limit or the level being played. The player in the UTG position is the first to act and that in the big-blind is the last, with turns being taken sequentially in a clockwise movement.

In the heads-up format the game is played by only 2 players, hence the regular table positioning structure not being possible to reproduce. There are only 2 positions considered in this format, and they are the button and the big-blind. As to preserve the order of play rules, which dictate that the big-blind must act last in the preflop stage, the player in the button posts the small-blind, being the first to act on the preflop stage of the game, and second on the remaining.

At the beginning of a hand, each player is dealt 2 hole cards, which only they can see. When it is their turn to act, they must choose to either bet, call or fold. When betting or calling, players are adding chips to the pot, the entirety of which will be collected by the winning player at the conclusion of the hand.

- **Bet –** Raise the current highest bet. When a player **Raises** or **Re-raises**, his action can be classified as a bet. If the player bets all of his chips, he is said to be **All-in**;

- **Call –** Match the current bet amount. Called a **Check** if the player does not need to put any more chips into the pot to do so, an event that can occur when he is positioned in the big-blind, or on the postflop stages of the game, when bets are yet to be placed;

- **Fold –** Forfeit the hole cards and any chances of receiving a portion of the pot.

There are 4 betting rounds in Texas Hold'em, namely the Preflop, Flop, Turn and River. For the betting rounds to progress, a bet must be called by at least one player, meaning that if one player bets and all the remaining fold, the betting player collects the pot and the hand ends.

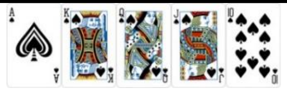
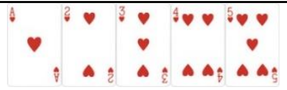
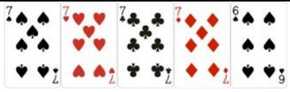
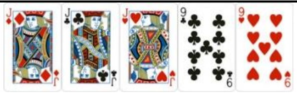
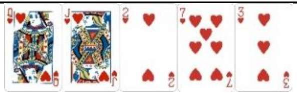
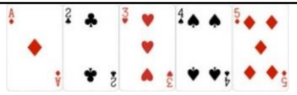
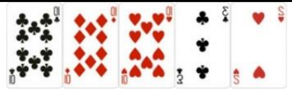
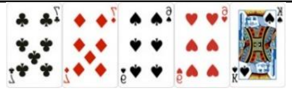
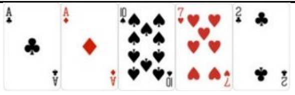
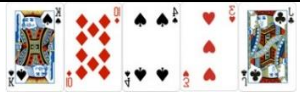
With the flop, turn and river rounds come the community cards - cards that are dealt face-up at the center of the table, constituting what is called "the board". The flop is composed of 3 community cards, while the turn and river are composed of 1 card each, making for a total of 5 community cards. Every player can use the community cards in combination with their hole cards, adding to the strength of their hand.

- **Preflop:** No community cards are revealed;
- **Postflop:**
    - **Flop:** The three first community cards;
    - **Turn:** The fourth community card;
    - **River:** The fifth and final community card.

Betting rounds are often referred to as "streets", and plays are usually separated between the preflop – when no community cards are revealed – and the postflop stages – the flop, turn and river. While the player in the UTG position is the first to act on the preflop stage, when the hand is first dealt, the small-blind acts first in all posterior streets which compose the postflop stages.

When all the betting rounds are over, the remaining players in the game show their hand and the winner is decided by evaluating the strength of each player's hand, composed by the 5 best cards in the play. Hand strengths are measured according to ranks, with each rank outperforming all the ones that precede it.

Table 2 – Texas Hold'em hand ranking

Rank strength	Rank description	Rank example
10	**Royal Flush:** A straight flush from Ace to 10.	
9	**Straight Flush:** A five card sequence of equal suit.	
8	**Four of a Kind:** A set of four cards of equal rank.	
7	**Full House:** Three of a kind plus a pair.	
6	**Flush:** A set of five cards of equal suit.	
5	**Straight:** A five card sequence.	
4	**Three of a Kind:** A set of three cards of equal rank.	
3	**Two Pair:** Two sets of two cards of the same rank.	
2	**Pair:** Two cards of the same rank.	
1	**High card:** The highest card of the five that make a hand, when no other rank applies.	

In the case of the hand being concluded through folding, the player who placed the bet inevitably wins the entire pot. If the hand reaches showdown, all the remaining players present their hand and the player with the best ranked hand is awarded the entirety of the pot. In the event of multiple players having a hand that is equally ranked, players' kicker card is evaluated, with the player having the highest kicker being declared the winner. The kicker card is the highest card in those available to the player – either in is hole cards or the flop – that does not combine to form one of the ranks presented above. For example, player A, holding A4, and player B, holding K4, go to showdown on a 44432 board. Player A will win the pot, because his fifth card in play is an Ace, which outperforms player B's King.

In case a tiebreak is not possible, the plot is split evenly amongst the players with the equally highest holdings.

2.2.2 Domain knowledge

Certain concepts of the game are fundamental to grasp the particularities and complexities of this study. The aspects deemed most relevant of some of these concepts are discussed in this chapter.

2.2.2.1 Table positioning

Table positioning is a key concept to this game, with players often referred to as being "in-position" or "out-of-position" according to their positional advantage. Players on the later positions have an information advantage since they already know the previous players' actions when their turn to act comes, and less players have a turn to act in front of them, and thus their play carries a lower amount of risk. While the UTG will look to play only premium holdings, the player in the button will be playing a wide range of hands when everyone folds before him since he has an incentive to win the pot, there are only 2 players left to act, and even if he gets called, he will be playing in-position for the remaining streets. This implies that players should ideally play a wider range of hands on tables with a lower number of opponents.

2.2.2.2 Blind-versus-blind dynamic

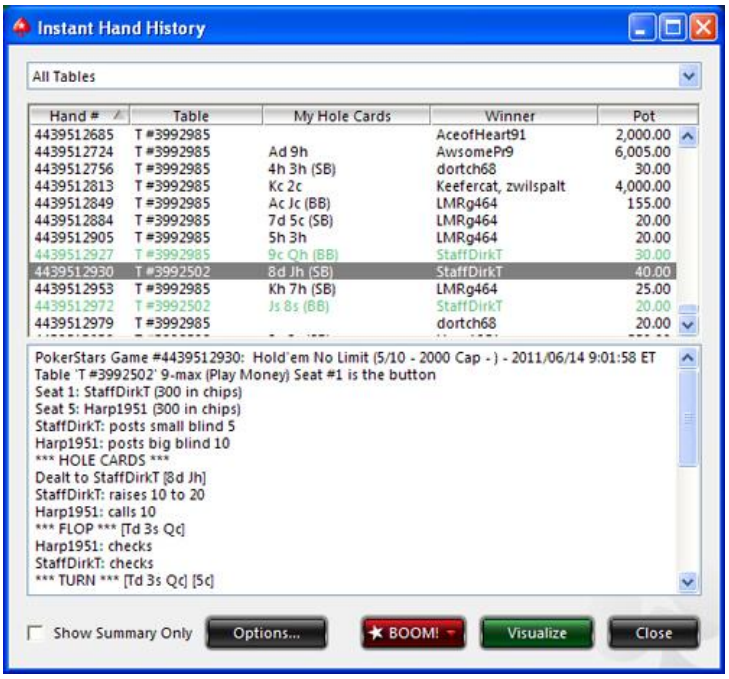
In the aforementioned hypothetical situation, the player on the button is said to be “stealing the blinds” when he raises with weaker holdings in order to try to take the pot. Players have an incentive to try to collect the blinds because they represent “dead money” in the pot – chips that were not voluntarily placed in the pot, and that no intervenient has a particular attachment to or incentive to defend. Due to pot odds, explained in a following chapter, this player can play profitably a wide array of hands in this situation. This fact in combination with the so called “blind-versus-blind” situations expose one of the most important dynamics of the game.

2.2.2.3 Hand nomenclature

Hands, or hole cards, are typically referred to by the combination of the cards’ ranks and its suit in descending order, with the rank as a capital letter and the suit as a lowercase letter. The ace of spades can be referenced to by ‘As’, for example.

2.2.2.4 Hand history representation

Hand histories are an historical log of a player’s hands, containing detailed information regarding the full setting of games the player participated in. This feature was made available in online Poker software, providing users with a mean of obtain an overview or retrospective of the hands played.



Hand #	Table	My Hole Cards	Winner	Pot
4439512685	T #3992985		AceofHeart91	2,000.00
4439512724	T #3992985	Ad 9h	AwsomePr9	6,005.00
4439512756	T #3992985	4h 3h (SB)	dortch68	30.00
4439512813	T #3992985	Kc 2c	Keefercat, zwilspalt	4,000.00
4439512849	T #3992985	Ac Jc (BB)	LMRg464	155.00
4439512884	T #3992985	7d 5c (SB)	LMRg464	20.00
4439512905	T #3992985	5h 3h	LMRg464	20.00
4439512927	T #3992985	9c Qh (BB)	StaffDirkT	30.00
4439512930	T #3992502	8d Jh (SB)	StaffDirkT	40.00
4439512953	T #3992985	Kh 7h (SB)	LMRg464	25.00
4439512972	T #3992502	Js 8s (BB)	StaffDirkT	20.00
4439512979	T #3992985		dortch68	20.00

```
PokerStars Game #4439512930:  Hold'em No Limit (5/10 - 2000 Cap - ) - 2011/06/14 9:01:58 ET
Table 'T #3992502' 9-max (Play Money) Seat #1 is the button
Seat 1: StaffDirkT (300 in chips)
Seat 5: Harp1951 (300 in chips)
StaffDirkT: posts small blind 5
Harp1951: posts big blind 10
*** HOLE CARDS ***
Dealt to StaffDirkT [8d Jh]
StaffDirkT: raises 10 to 20
Harp1951: calls 10
*** FLOP *** [Td 3s Qc]
Harp1951: checks
StaffDirkT: checks
*** TURN *** [Td 3s Qc] [5c]
```

Figure 4 – PokerStars hand history interpreter and reviewer

Widely used for statistical analysis of a player's own tendencies or those of his opponents, a standard format and syntax has been established to represent a play's stages or players' actions.

2.2.3 Pot odds

Pot odds are the ratio of the size of the pot to the size of a bet the player must call in order to continue playing. Suppose a hand reaches the flop which comes AhJh2s with 2 players remaining and a $15 pot. Player A bets $5, giving player B 4-to-1 (4:1) odds[7], since the total pot is now $20 and player B must pay a quarter-part of it in order to continue.

$$A_W * \mathrm{p} - A_L * (1 - \mathrm{p}) = 0 \qquad (1)$$

, where p is the probability of winning and A_W and A_L are the amounts won and lost when calling, respectively.

Player B needs at least a $4 \,/\, (4+1) \;=\; 0.8 \;=\; 20\%$ chance of winning in order to make a profitable call. In order to determine chances of winning and evaluate a winning probability,

[7] Pot odds are typically represented using a colon ':' (e.g. 4-to-1 represented as 4:1).

a player's number of outs is counted. If player B's cards are 7h6h he needs only one more card of the heart suit in order to make a flush draw. We know that 9 cards[8] of this suit remain in the deck out of 47 unknown cards[9]. Therefore, player B has a $\frac{9}{47} = 19.14\%$ chance of hitting a flush on the turn card, and a $\frac{9}{46} = 19.15\%$ chance of hitting it on the river. In total, this player has a $19.14\% + \left(\frac{9}{46} \times (1 - 19.14\%)\right) = 34.96\%$ chance of completing his hand until the end of the play. Since $34.96\% > 20\%$, player B is making the best mathematical decision by calling the bet.

In the given example, it is assumed that player B wins the hand every time he completes his flush draw, and the possibility of player A placing a bet on the turn is not taken into consideration. Considering that player A is an aggressive player who is likely to place a bet on the turn, it might appear that player B has no incentive to call since his chance of hitting the flush on the turn (19.14%) is marginally lower than required given the pot odds (20%). This, however, is not the case, due to implied odds (Campos, 2013) – odds that are inexistent but still considered due to being likely to win additional bets in case of hitting the hand, in this case directly related with player A's aggression tendencies.

The concept of blockers is also one of the most relevant for the decision-making process. If player A is holding a hand such as KcTd, he blocks his opponents possibilities of drawing a straight, gaining the ability of playing in a more passive manner, and possibly giving the opponent better odds, since he has access to private information that dictates that player B is less likely to hit a straight, if drawing for one.

2.2.4 Ranges

A range is a group of hands used to describe the most probable hands a player is believed to be playing in a particular context (Tipton, 2012). Ranges mirror a players strategy, and can be deduced from observation of a significant number of trials (2.2.8), assuming the player in analysis is playing according to a somewhat defined strategy. We can, for example, refer to the range a player might be opening[10] from the UTG position, or the range with which the big-blind will be defending from steal attempts from the button.

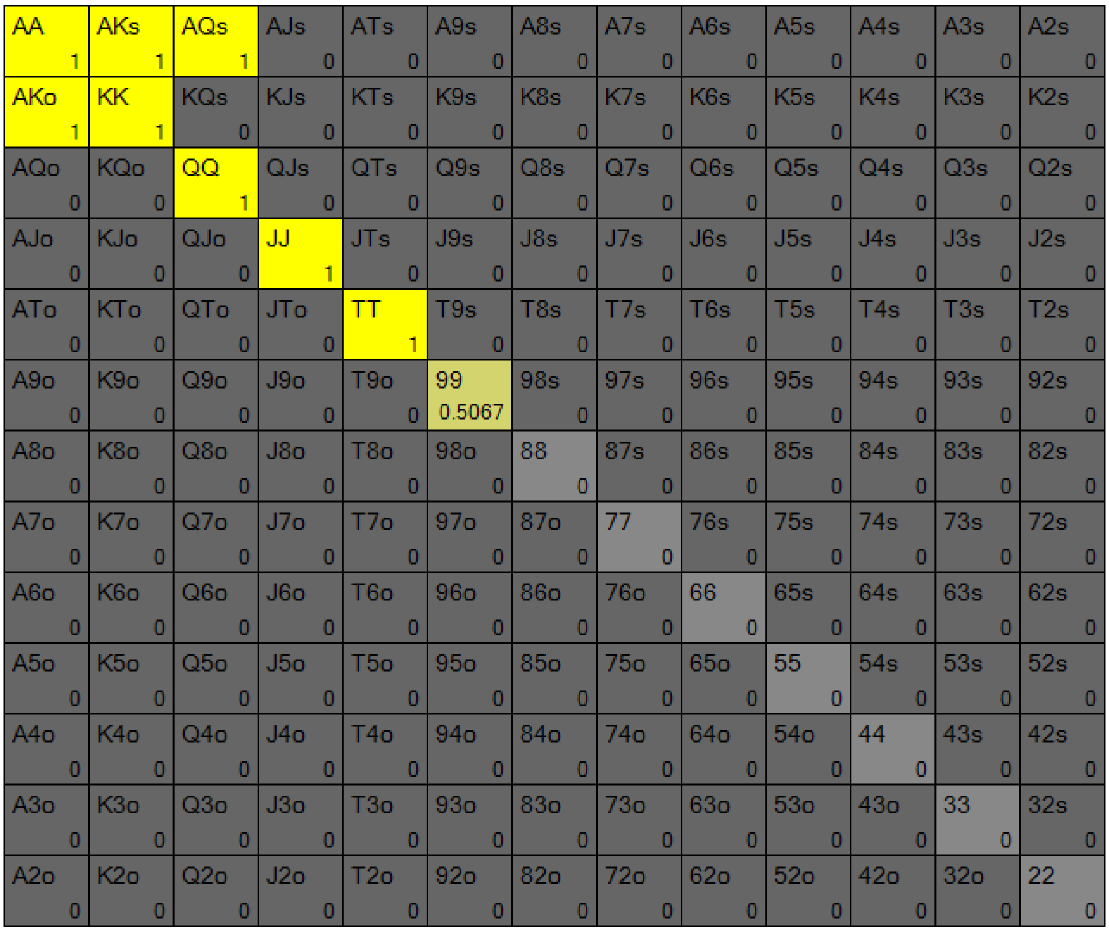
Hand ranges are typically abbreviated, being described in one of two formats:

[8] 13 cards minus the 4 heart suited cards in player's B hole cards and the flop.
[9] 52 cards minus player B's 2 hole cards and the 3 flop cards.
[10] Opening the pot is the act of placing the first raise in a given betting round.

- **Percentual:** Playing 3% of hands means the players range is the top portion of hands encompassed by the defined percentage – in this case, the top 3% of hands is composed by AA, KK, QQ, KK, TT, 99, AKs, AKo, AQs;
- **List:** The group of cards containing every ace, king, suited queen and offsuit jack combos with a kicker larger than 8 can be mentioned as {A2+, K2+, Q2s+, J8o+}.

In some situations, a range can be composed by a fraction of a particular hand. Suppose a player's strategy is to open 76s half of the time on the button and fold it the other half. His range is said to contain 76s with a weight of 0.5, or to contain 50% of all 76s hands. In the 3% range mentioned above as an example, 99 has a 0.5067 weight.

It is useful to be familiar with hand charts which are a representation of ranges, with the colored squares representing the holdings contained in the defined range, and the color opacity of each square's background representing the weight of a given hand.



AA 1	AKs 1	AQs 1	AJs 0	ATs 0	A9s 0	A8s 0	A7s 0	A6s 0	A5s 0	A4s 0	A3s 0	A2s 0
AKo 1	KK 1	KQs 0	KJs 0	KTs 0	K9s 0	K8s 0	K7s 0	K6s 0	K5s 0	K4s 0	K3s 0	K2s 0
AQo 0	KQo 0	QQ 1	QJs 0	QTs 0	Q9s 0	Q8s 0	Q7s 0	Q6s 0	Q5s 0	Q4s 0	Q3s 0	Q2s 0
AJo 0	KJo 0	QJo 0	JJ 1	JTs 0	J9s 0	J8s 0	J7s 0	J6s 0	J5s 0	J4s 0	J3s 0	J2s 0
ATo 0	KTo 0	QTo 0	JTo 0	TT 1	T9s 0	T8s 0	T7s 0	T6s 0	T5s 0	T4s 0	T3s 0	T2s 0
A9o 0	K9o 0	Q9o 0	J9o 0	T9o 0	99 0.5067	98s 0	97s 0	96s 0	95s 0	94s 0	93s 0	92s 0
A8o 0	K8o 0	Q8o 0	J8o 0	T8o 0	98o 0	88 0	87s 0	86s 0	85s 0	84s 0	83s 0	82s 0
A7o 0	K7o 0	Q7o 0	J7o 0	T7o 0	97o 0	87o 0	77 0	76s 0	75s 0	74s 0	73s 0	72s 0
A6o 0	K6o 0	Q6o 0	J6o 0	T6o 0	96o 0	86o 0	76o 0	66 0	65s 0	64s 0	63s 0	62s 0
A5o 0	K5o 0	Q5o 0	J5o 0	T5o 0	95o 0	85o 0	75o 0	65o 0	55 0	54s 0	53s 0	52s 0
A4o 0	K4o 0	Q4o 0	J4o 0	T4o 0	94o 0	84o 0	74o 0	64o 0	54o 0	44 0	43s 0	42s 0
A3o 0	K3o 0	Q3o 0	J3o 0	T3o 0	93o 0	83o 0	73o 0	63o 0	53o 0	43o 0	33 0	32s 0
A2o 0	K2o 0	Q2o 0	J2o 0	T2o 0	92o 0	82o 0	72o 0	62o 0	52o 0	42o 0	32o 0	22 0

Figure 5 - Hand chart with the top 3% hands

Understanding the concept of ranges is fundamental to the application of other strategic concepts, such as pot odds, equity and hand strength.

2.2.5 Hand evaluation

A player's decisions on the different betting streets of a play are controlled by the evaluation of his hand and strategy, as well as the opponents perceived strength. For this, it is imperative to have the ability to assess the quality of a hand – it's potential to be the best hand in play – at any point of the game.

Sklansky and Malmuth (Skalinsky and Malmuth, 1994) have classified pre-flop hands in human understandable and applicable terms by assigning them to separate groups, each of which having a defined strategy assessed based on the group's strength, player's position, number of players on the table and opponent classification (2.2.8).

A different approach that is found to have a strong correlation with the aforementioned methodology consists in simply ranking hands according to their chance of winning when going all-in preflop versus any random hand (Tipton, 2012), producing an approximated statistical measure of expectation for each hand through repeated randomized trials.

In order to generalize the ideas presented in (Skalinsky and Malmuth, 1994) and create an algorithm capable of evaluating a hand on any betting street, the state of the art methodologies analyzed in (Billings *et al.*, 1998; Teófilo, 2011) can be used.

2.2.5.1 Hand odds

The process of calculating hand odds involves enumerating all possible outcomes in order to assess the number of times each player is a winner or loser at the end of the play. It is considered that the opponents can have any of the remaining unknown cards – cards that aren't in hero's[11] hole cards or the community cards – and that both players will go to showdown. The hand ranks of all players are computed for every possible board and a comparison is drawn between them, evaluating the frequency with which each player will be ahead at the end of the play.

Concerns regarding this evaluation process revolve around the size of the sample space, which is proportional to the number of players in game and the number of unknown community cards remaining at the time of the evaluation. A heads-up game on the flop has 47 unknown cards (52 – 2 – 3) and a sequence size of 4 (the turn and river cards and villain's[7] cards), making for a total of $\frac{47!}{(47-4)!} \approx 4.28 \times 10^6$ card sequences, a quantity associated with a calculation time that can make the application of this methodology unfeasible.

[11] In Poker terminology, the player we are referring to when commenting a play is commonly referred to as "hero", and his opponent(s) as "villain".

Analyzing hand odds via a Monte Carlo simulation (Ulam, 1949) has been proven to be a viable solution (*Calculating Win Odds for Multiple Opponents - Poker for Programmers: Poker Algorithms and Tools for the C# Programmer*, 2006) – considering only a limited subset of the possible card sequences, a usable approximation is obtained with only a fraction of the calculation time necessary.

2.2.5.2 Hand strength

It is possible to assess the strength of a hand on every betting round through the use of enumeration methods in which the number of possible hands that are better, equal or worse than that being assessed is counted in order to retrieve a percentile ranking that is indicative of the hand's strength.

In contrast to the assessment described in the previous section, this is applicable on any betting street of the play and does not consider the remaining community cards. In comparison, this procedure is significantly less demanding, with a number of cycles that is not only much smaller – $C_2^{50} = 1225$ in the worst-case scenario – but is also not affected by the number of players being considered.

```
function HandStrength(hand, board) {
  var ahead = tied = behind = 0
  var handRank = Rank(cards, board)
  var opponentCards = CardCombos(deck - hand - board)

  foreach (opponentCards) {
    var opponentHandRank = Rank(opponentCards, board)
    if (handRank > opponentHandRank) ahead++
    else if (handRank == opponentHandRank) tied++
    else behind++
  }

  return (ahead + tied / 2) / (ahead + tied + behind)
}
```

Code snippet 1 – Evaluation of hand strength

This assessment can be extrapolated to multiway scenarios by raising its result to the number of active opponents.

$$HS_n = (HS_1)^n \qquad (3)$$

, where HS is the resulting hand strength and n is the number of active opponents.

2.2.5.3 Hand potential

Hand strength measurement provides an incomplete assessment of the overall quality of a hand, not accounting for the possibility of weaker hands developing into strong ones as the game progresses – in this case referred to as it's potential.

```
function HandPotential(hand, board) {
  var HP[3][3], HPTotal[3]
  var handRank = Rank(cards, board)
  var opponentCards = CardCombos(deck - hand - board)
  var simulatedBoards = EnumerateBoards(deck - hand - board)

  /* Consider all opponent cards */
  foreach (opponentCards) {
    var opponentHandRank = Rank(opponentCards, board)

    if (handRank > opponentHandRank) index = ahead
    else if (handRank = opponentHandRank) index = tied
    else index = behind HPTotal[index]++

    /* Consider all boards */
    foreach (simulatedBoards => simulatedBoard) {
      bestRank = Rank(cards, simulatedBoard)
      opponentBestRank = Rank(opponentCards, simulatedBoard)
      if (bestRank > opponentBestRank) HP[index][ahead]++
      else if (bestRank = opponentBestRank) HP[index][tied]++
      else HP[index][behind]++
    }
  }
  var PPot = (HP[behind][ahead] + HP[behind][tied]/2 + HP[tied][ahed]/2) /
(HPTotal[behind]+HPTotal[tied]/2)
  var NPot = (HP[ahead][behind] + HP[tied][behind]/2 + HP[ahead][tied]/2) /
(HPTotal[ahead]+HPTotal[tied]/2)
  return [PPot, NPot]
}
```

Code snippet 2 – Evaluation of hand potential

Evaluating a hand's potential involves analyzing how it's rank may evolve throughout the play as the community cards are dealt. Picture a scenario where hero is holding 5h4h on a Kh6s3h flop – with open-ended[12] straight and flush draws, he has a 59.98 % chance of winning and 1 % chance of tying versus two random cards, with any 2, 7, heart, 55 or 44 combo completing the hand and making it a winner. Despite this, the hand obtains a poor strength figure associated with it, since it is highly improbable for hero to have the winning hand were the game to end at this point. Thus, unlike the methodology presented for hand strength measurement, this assessment accounts for community cards and uses the

[12] Open-ended refers to a situation where a player has 4 cards to a straight that can be completed on either end (e.g. player has cards 3 to 6 with any 2 or 7 completing his straight).

previously described enumeration methods to calculate the positive and negative potentials of a hand.

- **Positive potential:** The probability of a hand improving when behind.
- **Negative potential:** The probability of a hand falling behind when ahead.

2.2.5.4 Effective hand strength

Combining the hand strength and hand potential algorithms, it is possible to measure the effective hand strength, which represents the probability of the hand being the best or improving to become the best.

$$EHS = HS_n + (1 - HS_n) \times PPot \tag{4}$$

, where HS_n is the hand strength and $PPot$ is the positive potential.

As denoted by Luís Teófilo’s study (Teófilo, 2011), this formula is deduced by setting the negative potential variable to 0. The full equation, representing the probability of winning, is as follows:

$$p_{win} = HS \times (1 - NPot) + (1 - HS) \times PPot \tag{5}$$

, where HS is the hand strength, $NPot$ is the negative potential and $PPot$ is the positive potential.

This method provides robust results that take every possible scenario into consideration. It does however have limitations associated with it, such as the assumption that opponents are playing any two cards with equal probability, as well as it’s inadequacy for application on the preflop and river betting rounds, due to the only available information being the player’s hole cards, and the hand being unable to evolve any further, respectively.

2.2.6 Expected value

The expected value (EV) of stochastic events is the weighted average of all possible values and the likelihood of each value. In Poker, it represents the average result of a given play obtained if it occurred thousands of times and is usually applied in relation to players actions in order to estimate the payout associated with it.

This figure refers to the amount of chips in the player's stack after making a decision, averaging over the uncontrollable and unknown information – community cards and opponents hole cards – that may affect the outcome of the play.

Disregarding loss leader type plays[13], the most profitable decision to make is always that with the largest EV.

The formula for calculating the value expectation of a decision can be formalized as follows:

$$EV = p_{win} \times (S + P) + (1 - p_{win}) \times (S + P) \tag{6}$$

, where p_{win} is the probability of winning, S is the player's stack size, P is the pot size.

Demonstrating the expected value estimation of a practical situation, consider a heads-up play scenario where both players start with a 30 BB stack. Villain places a 10 BB preflop raise which Hero calls and goes all-in on the flop for the remaining 20 BB. The pot has a total of 30 BB and Hero considers having a 40% chance of winning.

- If Hero folds, his stack at the end of the end will be 20 BB.
- If hero calls, his stack at the end of the hand will be $0.4 \times 60\ BB + 0.6 \times 0\ BB = 22.5\ BB$.

Since calling is the action with the largest expectation, it is the most profitable play.

When calculating the expected value of aggression plays, one can extend the formula to consider the fold equity the play employs over the opponent. This formula considers the chips gained by making the opponent fold, the chips gained when the opponent calls and Hero wins, and the chips lost when the opponent calls and wins.

$$EV = f_{fold} \times (S + P) + f_{call} \times e_{hero} \times (S + P) - f_{call} \times e_{villain} \times (S + P) \tag{7}$$

, where f represents the frequency of fold and call actions, e represents equities of both players, S is the player's stack size and P is the pot size.

Note that expert knowledge (Tipton, 2012) shows that calculating EV based on expected stack size is more advantageous than calculating it based on expected variation of stack size.

[13] A loss leader play in Poker is one that has a negative expected value but increases that of the general strategy, typically by making it less predictable.

2.2.7 Equity

Equity is the portion of the pot the player is entitled to would the hand end at a given moment, and is a function of hand strength (2.2.5.2). It is important to introduce the concepts of equity realization and denial in order to demonstrate how these may relate with strategic decision reasoning, and how the balance of actions in a strategy may be optimized in general and against particular opponents.

Similar to the concept of expected value but presented as a percentage, equity (or raw equity) is the average percentage of the pot a hand is expected to win when reaching showdown. Equity realization differs from raw equity by considering the action in the play's betting streets and comes into play in situations where players are forced to fold despite holding strong hands, becoming unable to realize their equity. The postflop dynamics in games such as No Limit Texas Hold'em provokes large intervals between the mathematical expectancy of a hand and it's true value, with factors such as the amount of time a player bluffs or fold to bluffs and the percentage of the pot a player bets holding a major influence.

The realization of equity can be measures as in $e_{realized} = e \times f_{realized}$ where e is the equity of a hand in a given situation and versus an opponents' range, and $f_{realized}$ is the frequency with which it is passible of realization.

Although pocket jacks has a 77.15% chance of winning versus a random holding, it is reasonable to estimate that it will only be able to realize it's equity about 60% of the time, considering that there is nearly a 45% chance of a broadway card coming on the flop and that the player will often find himself in a position where he has to fold against an aggressor. Thus, we say that this hand has a $0.6 \times 77.15\% = 46.5\%$ chance of realizing it's equity, or that it will only win 46.5% of the times it is played, unlike what the raw equity figure of 77.15% led us to believe.

Also important is to understand the concept of equity denial – the act of preventing a player from realizing his equity by forcing him to fold before showdown – and how it may impact betting strategies.

By denying opponents' equity, a player is gaining potential value. This is a common case study for continuation bets, with cases where it is mathematically viable to perform a continuation bet with nearly 100% of a range. In order to evaluate how much the opponent needs to fold for a bluff to be profitable, we calculate the required fold equity (RFE):

$$RFE = \frac{B}{B + P} \tag{8}$$

, where B represents the chip amount bet and P is the pot size.

2.2.8 Opponent modeling

Opponent modeling allows for the segregation of opponents according to characteristics of their strategy and skill level. While it may not be of major relevance in other strategic games where the performance lost by disregarding it is ignorable, it plays a key role in Poker agent's ability. The strategy often found to be the most profitable is not one based on constant, theoretically optimal decisions, but one that is based on modelling and responding to (exploring) the characteristics of the opponents and table environment.

By archiving and analyzing hands played against a specific opponent, it is possible to calculate an array of statistics that decompile his strategy:

- **VPIP – Voluntarily put $ in the pot:** The frequency in which the player calls on the preflop;
- **PFR – Preflop raise:** The frequency in which the player raises on the preflop betting round;
- **RFI – Raise first in:** The frequency in which player is the first player to raise in an unopened pot;
- **CBet – Continuation bet:** The frequency in which the player places a continuation bet after being an aggressor on the previous betting street;
- **Fold to 3bet:** The frequency in which the player folds to a 3bet after betting. Additional statistics similar to this is Fold to 4bet, 5bet (…);
- **XF – Check-fold:** The frequency in which the player folds to a bet after checking;
- **AF – Aggression factor:** Calculated based on the ratio of $\frac{bets}{calls}$ which is the frequency of aggressive moves in relation to passive moves.

All the presented statistics are a percentage between 0% and 100%, and are often decompiled (when applicable) in order to provide insights that are as closely related as possible to each scenario, based on the following factors:

- **Positional advantage:** The frequency in which the player took said action in-position (IP) or out-of-position (OOP).
- **Position:** The frequency in which player took said action in each of the table positions – UTG, MP, CO, BTN, SB, BB.
- **Betting street:** The frequency in which player took said action in each betting street – preflop, flop, turn, river.

Opponent models can be created based on these statistics according to pre-defined value intervals for which standardized adjustments to an agent's strategy can be induced. It is important to note that a significant sample is required in order to form elations based on these statistics, with these not being based on the number of hands played, but rather on the amount of opportunities of the action the statistic aims to represent (Patvs, 2013). On the event of the sample lacking significance, the agent should play a conserving strategy such as Nash equilibrium (0) or adjust based on the population tendencies.

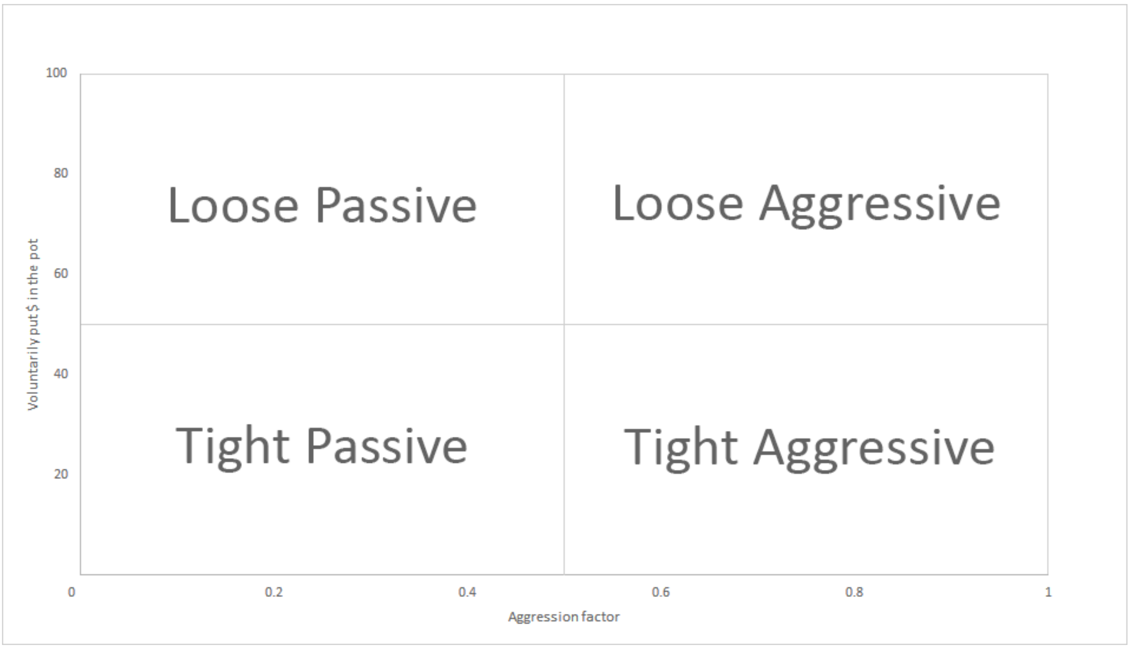
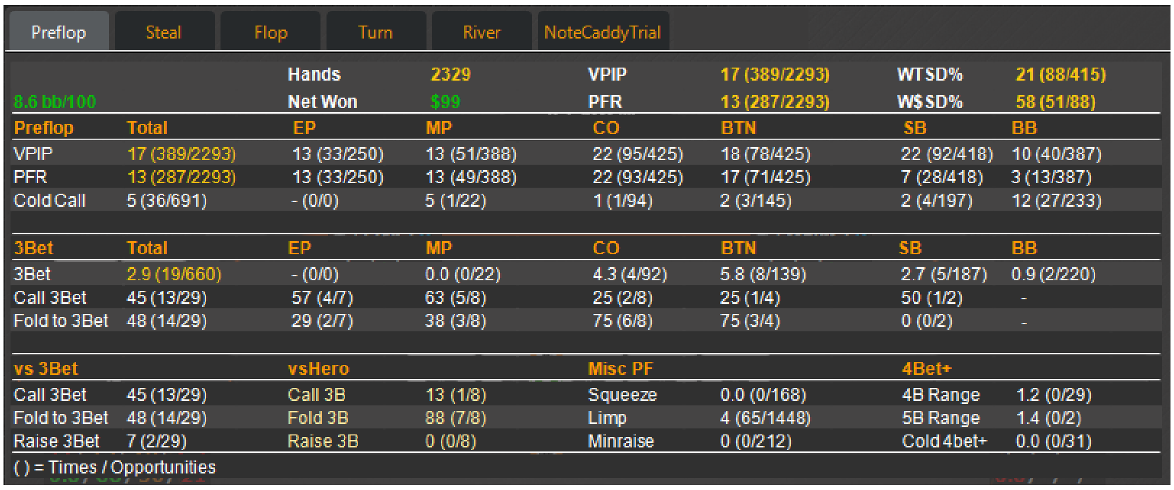


Figure 6 - Player classification

How the opponent model will influence betting strategy, which is primarily based on effective hand strength and pot odds. Hand strength assessment versus opponents perceived range.

With an agent's betting strategy being primarily influenced by hand strength and pot odds (Billings *et al.*, 1998) (see 2.2.5.4 and 2.2.3), one can improve by inference based on the opponent model, which provides the theoretical bases necessary to restrict an opponent's range in order to calculate hand strength taking into account the opponent's, as suggested in (Dinis Felix, 2008).

Software such as Hold'em Manager (*HM3 Official Page New Users*, 2020) and PokerTracker (*PokerTracker*, 2020) collect hand histories and calculate hundreds of statistics that are then shown in a heads-up display (HUD), aiding players in the decision-making process.



Preflop | Steal | Flop | Turn | River | NoteCaddyTrial

		Hands	2329	VPIP	17 (389/2293)	WTSD%	21 (88/415)
8.6 bb/100		Net Won	$99	PFR	13 (287/2293)	W$SD%	58 (51/88)
Preflop	Total	EP	MP	CO	BTN	SB	BB
VPIP	17 (389/2293)	13 (33/250)	13 (51/388)	22 (95/425)	18 (78/425)	22 (92/418)	10 (40/387)
PFR	13 (287/2293)	13 (33/250)	13 (49/388)	22 (93/425)	17 (71/425)	7 (28/418)	3 (13/387)
Cold Call	5 (36/691)	- (0/0)	5 (1/22)	1 (1/94)	2 (3/145)	2 (4/197)	12 (27/233)
3Bet	Total	EP	MP	CO	BTN	SB	BB
3Bet	2.9 (19/660)	- (0/0)	0.0 (0/22)	4.3 (4/92)	5.8 (8/139)	2.7 (5/187)	0.9 (2/220)
Call 3Bet	45 (13/29)	57 (4/7)	63 (5/8)	25 (2/8)	25 (1/4)	50 (1/2)	-
Fold to 3Bet	48 (14/29)	29 (2/7)	38 (3/8)	75 (6/8)	75 (3/4)	0 (0/2)	-
vs 3Bet		vsHero		Misc PF		4Bet+	
Call 3Bet	45 (13/29)	Call 3B	13 (1/8)	Squeeze	0.0 (0/168)	4B Range	1.2 (0/29)
Fold to 3Bet	48 (14/29)	Fold 3B	88 (7/8)	Limp	4 (65/1448)	5B Range	1.4 (0/2)
Raise 3Bet	7 (2/29)	Raise 3B	0 (0/8)	Minraise	0 (0/212)	Cold 4bet+	0.0 (0/31)

( ) = Times / Opportunities

Figure 7 - Example Hold’em Manager HUD statistics pop-up

2.3 Theoretical concepts applied to Texas Hold’em

Texas Hold’em can be attributed the following classifications:

- **Incomplete information:** Players don't have knowledge of all previous events, specifically the initialization event, and are therefore unaware of other players' hole cards;
- **Non cooperative:** No third-party enforceable contracts are modeled in the game, and contracts are unable to be self-enforced. Information sharing is considered a form of cooperation termed “collusion” and is against the game's rules;
- **Finitely long:** The winner and payoff is unknown until all the moves are completed;
- **Discrete:** A finite number of elements compose the game's environment;
- **Zero-sum:** A player's winnings equal other players' losses, unless the game has rake associated, which is common in games hosted in casinos;
- **Stochastic:** Responses to player's actions are influenced by chance events;
- **Sequential:** Players are aware of previous players' actions;
- **Asymmetrical:** The payoff and overall course of the game is influenced by a player's amount of chips and would in many scenarios be impossible of matching by switching 2 players.

This game is classified as sequential, and therefore extensive form will be the utilized form of representation in this study, adhering to the norm which implies that normal form shall be used to represent simultaneous games, and extensive form to represent sequential ones.

The characteristics of the various Poker variants can in most cases be refined further based on the betting structure in place, of which there are the following for Texas Hold'em:

- **No Limit (NL) –** There is a minimum bet size for bets and raises, but no upper limit. A player can bet the full amount of his stack ("go all-in") and re-raise (3bet, 4bet, …) as many times as his stack allows at any given betting round.

- **Fixed Limit (FL) –** As the name implies, bets and raises can only be placed in fixed amounts, with 2 defined bet sizes in the game: small bet and big bet. Players can only place small bets on the preflop and flop betting rounds, and big bets in the remaining. The big-blind is equal to 1 small bet. The first bet in any round must be exactly equal to the small bet size, if on the preflop or flop, or the big bet size, on the turn and river. Raises must also exactly equal to the relevant bet size and a total of 3 raises can be places in any given betting round.

- **Pot Limit (PL) –** Similar to No Limit but the maximum amount a player can bet or raise is determined by the size of the current pot, which includes not only the bets made in the current and previous betting rounds, but also the amount one has to call in order to match any previous bets $(pot = pot + current\ round + call)$. This is one of the more complicated betting structures and is particularly popular in the Omaha variant.

- **Spread Limit –** A compromise between the No Limit and Fixed Limit structures, with bets and raises being capped but raises not necessarily matching the relevant bet amount, being instead defined by a spread.

- **Cap Limit –** Similar to No Limit but there is a limit to how much a player can bet in a given hand.

This study intends to focus particularly on the No Limit and Fixed Limit betting structures, with the first of these being the one which experiences the most popularity, having surpassed the latter in recent years.

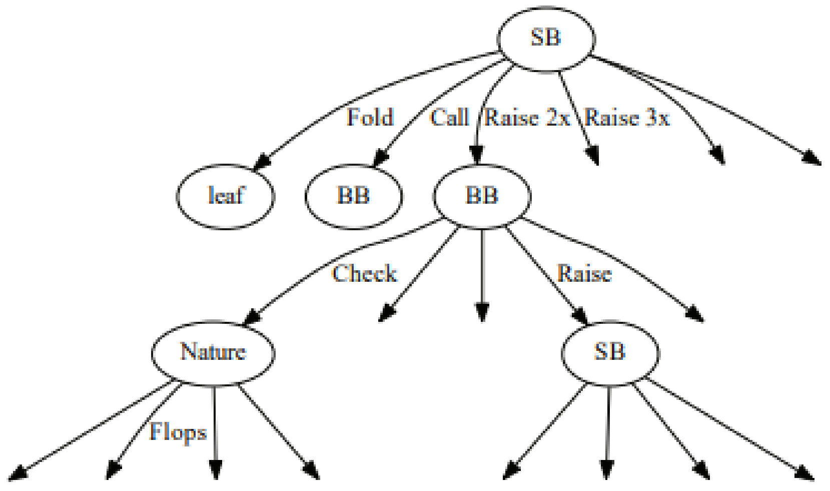


Figure 8 – Extended decision tree representing the No Limit Texas Hold'em game in extensive form

Analyzing the extensive-form representations of games in both variants, it is observable that the Fixed Limit structure produces decision trees which have a smaller maximum possible distance between the root and leaf, and each decision point is significantly simpler. The reasons for such observations are directly deductible:

- The existence of a total amount of raises per betting round impacts the maximum possible distance between the root and leaf;
- The fixed bet and raise amount impacts decision points' available actions.

2.3.1 Approximate games

When games are too large and computationally expensive to solve, an approximation to the full solution is often found to be the only viable methodology for studying certain aspects of the game. Although incomplete, approximate solutions are still capable of providing a deep enough overview of a game or play that humans can extract information from and adapt or add to their strategies.

The full game of Heads-up No Limit Hold'em is an example of a game deemed too large to solve, and it is easy to understand why. Depict a scenario where both players have 100 BB stacks and the play starts at the small-blind, who can place an open-raise of any size, which can then be re-raised by the big-blind. This cycle can repeat itself until one of the players decides to either call or fold, and if they call, one of 22 100 flops is dealt, and this cycle restarts until showdown is reached or one of the player folds. Considering that each of the individual action combinations that might occur in the game need to be calculated independently, one can imagine the size of the resulting game tree. Will Tipton (2012) estimates a full solution to game described to occupy 5 Petabytes of disk storage space, and even this is an approximate solution considering the concessions made in order to simplify calculations.

For the described reasons, games are approximated in order to make them manageable from a computational perspective, as well as to enable human players to extract patterns from such studies which they can apply to their games, something that isn't feasible by studying the full solution. More so, removing elements from the game based on theoretically correct assumptions does not degrade the quality of the solution to a degree which affects the best grasp a human player can possibly obtain from studying them.

2.3.2 Abstractions

Classifying and limiting strategy sets – or the amount of possible actions for a player – is an important task in the field of applied game theory, where expert knowledge and other various optimizations can be utilized in order to limit the strategy space, simplifying the game and making solution calculation feasible.

In this sense, ignoring certain elements of a game in order to limit the size of a solution is termed an abstraction. Through the identification of player's tendencies or game situations where players are strongly incentivized to select a certain subset of their available actions, the scenarios which are believed to be unlikely can sometimes be ignored with no harm to the quality of the solution.

There are 3 primary approximation methods for the game of Heads Up No Limit Hold'em (Tipton, 2012):

- **Narrow players' strategic options:** Lines of action which are nearly always incorrect or highly uncommon can be excluded;
- **Remove future card possibilities:** Instead of considering all the possible cards that might be dealt on the flop, turn and river betting rounds, a significant-enough subset of the possible cards can be considered in order to limit the simulation while not affecting predictions. For similar reasons, a play can be forced to finalize on a certain street through an all-in or fold action, disregarding any subsequent action;
- **Consider only part of a hand:** In order to study a play on a late betting street such as the river, any previous play needs not to be considered apart from each player's starting hand range.

These methods combine multiple abstractions which can be decoupled into the following factors:

- **Ranges:** Player's ranges are capped according to the statistical model attributed or the situation in play, considering only a fraction of both player's entire sample space of possible hole cards;

- **Stacks sizes:** Player's strategic options are proportional to their stack sizes, and therefore a simplification of this characteristic of a given game state may be conducted;

- **Bet sizes:** It is assumed that players only utilize 3 bet sizes for any given situation, simplifying the solution to a major degree, specifically in games with a No Limit betting structure;

- **Bucketing:** This technique consists in considering a meaningful subset of flops instead of the 22 100 available possibilities, dramatically reducing the requirements for the calculation of this betting street. Will Tipton (2012) documents a functional bucketing methodology, with PioSOLVER and MonkerSolver[14] having allegedly improved upon this method for optimized performance and results, the first of which claims to perform a simplification to only 1 755 flops, showing a 92% reduction (*Choosing a subset of flops to represent the whole game – PioSOLVER*, 2020);

- **Texture abstraction:** This technique is similar to that of bucketing, but instead of applying to flops applies to card suits instead, reducing the number of board possibilities that would otherwise have to be calculated.

2.3.3 Game theory optimal

Game theory optimal or "GTO" is a term used in the universe of Poker to describe an optimal, unexploitable strategy – a strategy that is in the Nash equilibrium.

In order to understand the concept of an optimal strategy in theoretic terms, we first have to understand what a strategy is in the context of the game of Poker – a strategy is a function that returns an action given a past sequence of events and a hand, mathematically described by $A = f(E, H)$ (Alex, 2014). While some strategies are easy to write down and follow, for which push/fold charts are an example, others aren't feasible of being stored without abstractions to the existing information or the set of possible actions from which to choose from – in the example of push/fold charts, an abstraction to the sequence of past events E is made, rendering it binary - whether or not an opponent went all-in before us.

A Nash equilibrium is obtained when both of the following conditions are met:

[14] PioSOLVER and MonkerSolver are game analysis and simulation software which are discussed in depth in chapter 3.2 of this study.

1. Each player has perfect knowledge of every other player's strategies;
2. Each player plays a strategy that maximizes their utility, given the opponents' strategies;

A simple example can be found in the game of rock-paper-scissors, where it can be said that a player employs a Nash equilibrium strategy if he consistently picks one of the 3 possible actions at random, effectively preventing his opponent from exploiting his strategy. In this scenario, the best strategy the opponent can employ is to play the exact same strategy, and thus, the Nash equilibrium is said to have been reached. Any deviation the opponent makes from this strategy leads to a decrease in his utility.

Nash equilibrium strategies for the game of Poker are attractive for a number of reasons:

- They guarantee a certain amount of utility;
- They simplify the game by consistently making the same decision instead of adapting to the opponent's strategy or the action history of the current state of the game;

However, these also have their pitfalls – in the scenario of playing against a weak opponent who makes a lot of mistakes, employing a Nash equilibrium strategy reduces the expected value a better player expects to achieve from playing this game, as this strategy does not exploit the opponent's mistakes.

Porting the rock-paper-scissors example to the game of Poker, how can we know if a set of strategies is within the Nash equilibrium in a heads-up game? Provided we are the player in the Small Blind and have perfect knowledge regarding the range of hands the player in the Big Blind is calling our shoves with, can we increase the expected value of our play by folding any of the hands we are shoving, or shoving any of the hands we are folding? What about if we apply the same process to the scenario in which we are the player in the Big Blind? Provided we can't increase our expected value in any of the aforementioned scenarios, it can be said that the strategy pair is within the Nash equilibrium.

While the Nash equilibrium guarantees un-exploitability in 2 player games such as Heads Up No Limit Texas Hold'em, this concept cannot be ported to games with 3 or more players, which have additional factors to be considered such as implicit and explicit collusion and the fact that a player can unilaterally gain or lose utility by modifying his strategy.

2.4 Approaches

This chapter focuses on game theoretical solution concepts, which dictate formal rules for predicting how a game will play out. The achieved solutions – also termed predictions – describe the players' strategies and the result of the game, and are achieved through the application of multiple different concepts, with the most commonly used being those based on equilibrium, and most famously the Nash equilibrium (2.1.4).

In the following sections, some of the approaches that have previously been utilized in order to create autonomous Poker playing agents will be documented.

2.4.1 Rule based

A rule-based approach is the most intuitive methodology of developing autonomous agents, basing their behavior on conditional rules. This implies that the agent is incapable of adapting and easy to predict after being observed by a period which may vary according to the complexity and number of rules.

Besides the immediately obvious limitations, this approach presents other drawbacks when applied to the game of Poker, such as the fact that it requires expert knowledge, the difficulty to represent decisions made based on player's intuition and the size of the full game making it too difficult to meaningfully represent in such manner.

2.4.2 Evolution based

Evolutionary game theory (EGT) originated in 1973 (J. Maynard Smith, 1973) and is based on the application of game theory to evolving populations in biology, differing from classical game theory by placing extra focus on the dynamics of strategy change.

According to EGT, evolution is modelled as behavioral linkages between the parents and offspring, with these principles having been previously applied to the context of Poker. This methodology is based on the creation of a competition for space in a population, whose members need to learn which strategies are successful in order to propagate them into further generations (Bartone and While, 2000).

Implicit evolutionary learning processes such as this provide adaptive capacities to the agents developed since this is an essential trait that enables them to learn and exploit opponent's weaknesses, maximizing profit.

2.4.3 Simulation based

Simulation based approaches refer to the Monte Carlo Simulation (Ulam, 1949) and are based on enumerating the entire sample space of possible outcomes, or at least a significant portion of it, through randomized trials, with the intention of empirically obtaining best response for a given game state.

Such simulations rely on a large number of simulations to the opponent’s behavior, which uses a large number of iterations and can produce inaccurate predictions when applied to games of incomplete information (Teófilo, 2016). In order to overcome this, pre-established profiles that have been previously attributed to the opponent can bias the sampling and drastically reduce the number of iterations, as well as produce more accurate results.

The most relevant simulation-based application to the development of autonomous Poker playing agents is the Monte-Carlo Tree Search algorithm (MCTS), which is a simulation-based algorithm adapted for sequential products. This algorithm initializes the game tree by creating a single root node containing the game state, and iterating through the following stages a fixed number of times:

1. **Selection:** selection of a leaf node to expand;
2. **Expansion:** add child nodes to the selected node;
3. **Simulation:** simulate on all the nodes added in the previous step until the leaf is reached.
4. **Backpropagation:** the value resulting from a given simulation’s followed path is stored on the nodes that define it.

2.4.4 Equilibrium based

Although there are equilibrium algorithms other than Nash’s, we will only focus on it due to the popularity and use-cases demonstrated for the specific use-cases of this study.

As defined on chapter 2.1.4, a Nash Equilibrium is said to be reached when the game participants, with a set of mixed strategies, cannot improve their performance by unilaterally changing their strategy. This set of strategies assumes, however, that the players always make the best possible move, which does not hold true for the game of Poker, where even experienced players may find themselves in situations where they are uncertain of the optimal play.

It must also be emphasized that a Nash-Equilibrium strategy won’t, in many cases, be the most profitable strategy, as it does not exploit the opponent. It does, however, ensure that an agent’s utility won’t fall bellow a certain threshold, which is a significant and usable achievement for the world of Poker in itself.

The equilibrium which can be computed is based on abstractions such as those described on section 2.3.2, achieving an Epsilum-Nash equilibrium which must posteriorly be translated into the full game, at the expense of degradation and added exploitability to the solution, as it is dependent on the used abstraction (Teófilo, 2016).

2.4.5 Counterfactual Regret Minimization

The Counterfactual Regret Minimization (CFR) is a self-play algorithm intended to find approximate Nash-Equilibrium solutions for sequential games larger than those that can be computed through traditional linear programming techniques.

The algorithm starts off with a random strategy, playing every action at every decision point with equal probability. By simulating games against itself and revisiting its decisions, the strategy is improved over billions of iterations, coming consistently closer to the optimal strategy.

Based on the concept of counterfactual regret initially defined by Zinkevich et al. (2009), summing the total amount of regret for each decision point is how the algorithm improves, where regret signifies how much better the algorithm would have done over all of the simulated games if it had always played a certain action at a certain decision point. An action with a positive regret value is an action one should opt for, as it is the one which offers the most value (Johanson, 2015).

3 State of the Art

This chapter intends to provide an in-depth depiction of the current state of the field of studies of game-theory applied to the game of Poker. Section 3.1 provides an overview of the academic efforts devoted to this thesis' field of studies, while section 3.2 dives into commercial software which is of interest for this study.

3.1 Academic Poker studies

Due to the study of games being a major driving force in the field of artificial intelligence, and with games such as Checkers and Chess being already solved, academics placed their focus on the game of Poker. Endorsed by an ever-growing number of institutions, annual poker-bot competitions are held and universities have departments dedicated specifically to the study of this game.

Some of the most renowned competitions are:

- **MIT Pokerbots –** Computerized poker tournament where teams have one month to develop a completely autonomous poker agent to compete against other teams (MIT, 2021);

- **Brains Vs. AI –** Matches between a group of selected professional Poker players, specializing in the Heads Up No Limit Hold'em variant, and an artificial intelligence algorithm developed by Carnegie Mellon University (Carnegie Mellon, 2017). Taking place in 2017, the AI beat poker players with statistical significance over a sample of 120 000 hands (PokerNews, 2017);

- **Annual Computer Poker Competition (ACPC) –** Competition held annually at the AAAI[15], attracting enthusiasts from all over the world and featuring separate competitions for the games of Limit Texas Hold'em for 2 and 3 players, Heads-up No Limit Texas Hold'em and Kuhn Poker for 3 players (CRPG, 2020).

Groups which are the most prominent in the artificial intelligence applied to the game of Poker field of studies are:

- **Computer Poker Research Group (CPRG) –** Based on the University of Alberta, this group has laid the groundwork for the field of academic Poker research has achieved milestones regarding the study of the game of Poker since 1997 and is responsible for many of the most renowned Poker bots on record. It's most recent prominent development is the DeepStack.AI poker bot, discussed in chapter **Error! Reference source not found.** (Dr. Michael Bowling, Trevor Davis, Dustin Morrill, 2020);

- **Carnegie Mellon University (CMU) –** Already mentioned in regards to the Brains Vs. AI event, many of the breakthroughs seen in this field of studies are associated with this university, with the most recent prominent developments being the Libratus and Pluribus bots, discussed in chapters 3.1.4 and 3.1.6, respectively;

The following sections regard some of the most renowned autonomous Poker playing algorithms, with details as to the methodologies and processes implemented and the results obtained, allowing the reader to grasp the evolution process such algorithms suffered over the years, with research groups and competitions such as the ones mentioned being a major driver, pushing those involved in this field of studies to break barriers, re-iterate and optimize year after year.

3.1.1 Polaris (2007)

Polaris was developed by University of Alberta's Computer Poker Research Group to play the Limit variant of the Texas Hold'em game, only playing the Heads-up format. Being composed of a number of previously developed bots, this bot contains an array of strategies from which it chooses during play, and had been in development for 16 years by the time it was put to the test in a match against professional Poker players in 2007. The learning algorithm of this bot can be categorized as counterfactual regret minimization (2.4.5) and restricted Nash response (2.1.4).

In this competition, Polaris played four matches, composed of 500 hands each, against professionals Phil Laak and Ali Eslami. While this is a lackluster sample, a variance reducing

[15] AAAI – Conference on Artificial Intelligence - https://www.aaai.org/

method was put in play where the player and the bot received the same cards in an alternate fashion, effectively meaning that the overall hand strength of the hands received by both the player and the bot was the same by the end of the match (Glaister, 2007). Polaris record in this event was one win, one tie and two losses.

In 2008, the Second Man-Machine Poker Championship was held with six professional players facing Polaris, in an event where the bot came ahead with three wins, two losses and one tie (Alberta, 2008).

At the time, Polaris' results were ground-breaking, effectively proving that artificial intelligence can stand a match against even the most trained human opponents.

3.1.2 Claudico (2014)

Claudico is a Poker-playing bot developed by Tuomas Sandholm[16] and his students in the Carnegie Mellon University. The name is derived from the latin expression "I limp", a strategy which the bot often employs (Malara, 2015).

This bot was directed at the No Limit variant of the Texas Hold'em game, in the Heads-up format, and required a supercomputer with 16 terabytes of RAM to complete. Despite the significant computational requirements, this bot was considered innovative due to the abstractions applied to the No Limit variant. It's learning algorithm falls under evolutionary game theory (2.4.2), with reports of it's application to equilibrium solution learning in order to accelerate the convergence process of the solution, by breeding and merging different equilibrium profiles (Teófilo, 2016).

In the 2015 Brains Vs. AI competition, Claudico faced four of the world's best Poker players at the time – Doug Polk, Dong Kim, Bjorn Li and Jason Les – in a total of four matches lasting 20 000 hands and 14 days, including one rest day in the middle. A variance reduction mechanism such as the one described in 3.1.1 was applied. The 80 000 hand sample of human-computer play which resulted was the largest existing to this date.

The final result of the match was determined by the overall chip count after all hands were played, considering that the four human players were playing as a team against the bot. The bot ultimately lost by a disadvantage of 732 713 chips, representing 36 buy-ins at the stake played – a figure considered to be of statistical significance, preventing the result of the match from being a draw and providing the human team with the victory (Glatzer, 2015).

[16] Tuomas Sandholm is a Computer Science professor at the Carnegie Mellon University who has demonstrated a great amount of interest in the research of artificial intelligence applied to the game of Poker (Sandholm, 2020).

3.1.3 Cepheus (2015)

Developed by the Computer Poker Research Group at the University of Alberta, Cepheus is reported to be the first Poker bot to have weakly solved the game of Heads-up Limit Texas Hold'em.

In the January 2015's paper entitles "Heads-up limit hold'em poker is solved", published in Science Maganize, it is shown that an optimal counter-strategy to Cepheus is only expected to win 0.000986 big blinds per game, a value of magnitudes which enable the authors – Michael Bowling, Neil Burch, Michael Johanson and Oskari Tammelin – to claim that the game is, indeed, "essentially" solved (Bowling *et al.*, 2015).

Using the CFR+ learning algorithm, which is an optimized variant of counterfactual regret minimization (2.4.5), Cepheus solved the game without resorting to abstraction, achieving a near-zero exploitability, rendering it unbeatable in the long-run (Teófilo, 2016).

CRPG provides a website[17] where users are able to play against the bot.

3.1.4 DeepStack.AI (2016)

DeepStack.AI is claimed by it's authors – the Computer Poker Research Group at University of Alberta – to be the first artificial intelligence algorithm to beat professional Poker players at the game on Heads-up No Limit Texas Hold'em (CPRG, 2016).

In a study completed December 2016 and published by Science Maganize in March 2017, involving 44 000 hands against 11 professional Poker players, DeepStack won 49 big blinds per 100 hands – a small sample, but also one with a win rate so high that it confirms the author's statement (Moravčík *et al.*, 2017).

While using abstraction ideas, this agent uses a fundamentally different approach from abstraction-based algorithms, restricting the number of actions in it's lookahead tree and continually re-solving the public game state, which in combination with heuristic search – in a first of it's kind application for imperfect information games – is responsible for the success of this algorithm's approach.

3.1.5 Libratus (2017)

Libratus is Claudico's successor, also directed at the No Limit variant of Heads-up Texas Hold'em, but intended to be of more general application, outside of the context of the game of Poker.

[17] Play against Cepheus in at http://poker-play.srv.ualberta.ca/.

Developed by Carnegie Mellon University, Libratus uses the CFR+ algorithm, a state of the art algorithm unveiled in the Cepheus agent, and didn't reduce the computational requirements to be ran in comparison to is predecessor, taking more than 15 million core hours of computation to be built, in comparison to just 3 million for it's previous iteration (Hsu, 2017).

In January 2017, Libratus was matched against four professional Poker players, in a competition where the total number of hands to be played increased to 120 000 in order to assure the statistical significance of the results (Spice and Allen, 2017). The four players were grouped into two teams, and the variance reduction mechanism described in 3.1.1 and 3.1.2 was applied. Unlike it's predecessor, Libratus beat not only both teams, but all four players, by a statistically significant margin of 14.7 big blinds per 100 hands – a win-rate which is considered to be very high, which in a sample as large as this event's proves without a doubt that this is a winning algorithm.

3.1.6 Pluribus (2019)

Developed by Facebook's AI Lab and Carnegie Mellon University, Pluribus became the first agent to beat humans outside a Heads-up setting in a the game on No Limit Texas Hold'em. Directed at the 6-max format, in which 6 players sit at the table, Pluribus was tested with statistical significance against professional Poker players, including two WSOP[18] winners. Being put to the test on both a "five Ais + one human player" and "five human players + one AI" setting, the agent won an average of 5 chips per hand in the latter setting.

Pluribus' success if a product of capitalizing upon the knowledge provided by Carnegie Mellon's studies already put to the test in the Libratus agent and innovating upon it. Incorporating "a new online search algorithm that can efficiently evaluate its options by searching just a few moves ahead rather than only to the end of the game" and "new, faster self-play algorithms for games with hidden information", this agent improves the CFR+ algorithm, providing it not only with improved in-game performance, but also improved computational performance, making this break-through available using $150 of cloud computing resources, in contrast to the millions of dollars used in computer equipment for the developments described in the chapters dedicated to Claudico, and it's successor, Libratus (Brown, 2019).

3.2 Commercial game solvers

Surfacing on the market recently, a game solver is a software which allows a player to insert a game's state, utilizing machine learning to compute the best possible strategy. Through

[18] WSOP – World Championship of Online Poker

the usage of data aggregation and visualization techniques, this strategy is then packaged into a format the user can interpret and study.

With a rise in the game's competitiveness, a growing number of players started resorting to such software, and solvers now constitute well established studying tools which are in every serious player's toolbox.

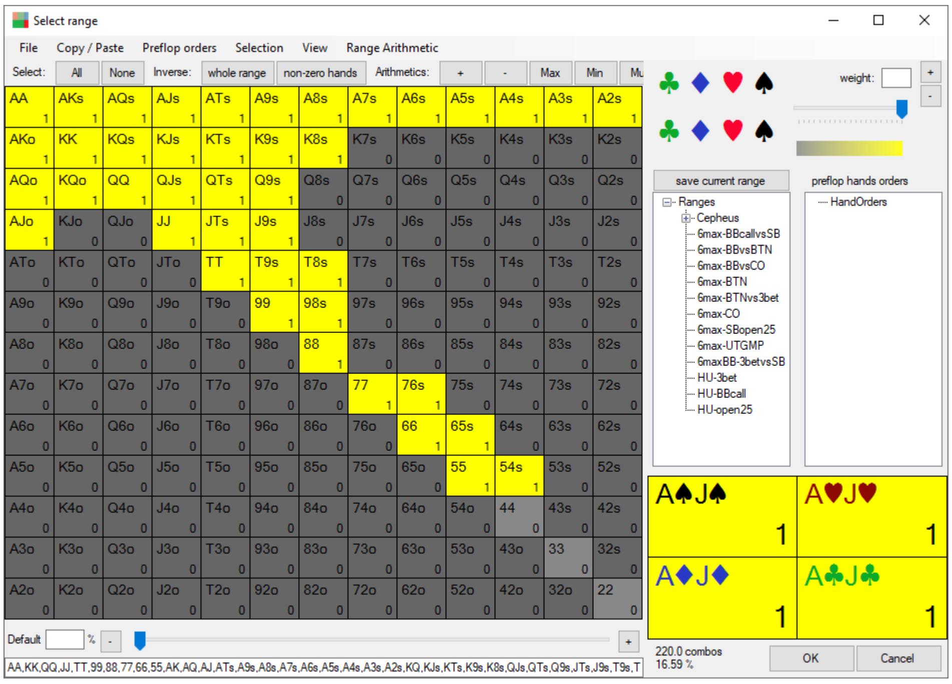
3.2.1 PioSOLVER

Surfacing on the market in the year of 2015, PioSOLVER (*Piosolver.com - PioSOLVER*, 2020) established itself as the de-facto standard of endgame solving (Ganzfried and Sandholm, 2015) software for the game of Hold'em. Utilizing user input to create a game scenario, this software demonstrated to be capable of formulating game theory optimal strategies with a high degree of accuracy, allowing players to analyze the game in an analytical manner to a degree which that was uncommon up to this point.

With most of the player's study of the game at the time consisting of equity simulations in characteristic range versus range scenarios, PioSOLVER was the first product to provide players with human-interpretable solution trees from which they could derive information that they could implement into their strategy, potentially covering their leaks while also learning how to optimally adapt to and exploit their opponents.

In order to analyze a game situation, users are required to input the players' stack sizes, bet sizes and ranges. While the first information is deterministic, users are required to make correct assumptions for the remaining in order to obtain an accurate solution. Although estimation errors in the arbitrary game situation can degrade or altogether invalidate the quality of the solution, it is important to note that human players tend to use standardized bet sizes, and a player's range in a certain situation can be approximated with a significant enough sample.

Demonstrating how to use this software to study a situation that consists of a 6-Max No Limit Hold'em flop spot. The pre-flop action folded to the button who open-raises to 2.5 BB, the small-blind re-raises to 8.5 BB, the big-blind folds and the player on the button calls, revealing a Qs Jh 2h flop. In order to generate a solution for the flop, turn and river sub-games, we start by make reasonable assumptions for both player's ranges. It is important to note that since we intend to study the post-flop stage of the game, the player in the small-blind is considered to be the out-of-position player, even though he has the positional advantage in the pre-flop stage of the game.



AA 1	AKs 1	AQs 1	AJs 1	ATs 1	A9s 1	A8s 1	A7s 1	A6s 1	A5s 1	A4s 1	A3s 1	A2s 1
AKo 1	KK 1	KQs 1	KJs 1	KTs 1	K9s 1	K8s 1	K7s 0	K6s 0	K5s 0	K4s 0	K3s 0	K2s 0
AQo 1	KQo 1	QQ 1	QJs 1	QTs 1	Q9s 1	Q8s 0	Q7s 0	Q6s 0	Q5s 0	Q4s 0	Q3s 0	Q2s 0
AJo 1	KJo 0	QJo 0	JJ 1	JTs 1	J9s 1	J8s 0	J7s 0	J6s 0	J5s 0	J4s 0	J3s 0	J2s 0
ATo 0	KTo 0	QTo 0	JTo 0	TT 1	T9s 1	T8s 1	T7s 0	T6s 0	T5s 0	T4s 0	T3s 0	T2s 0
A9o 0	K9o 0	Q9o 0	J9o 0	T9o 0	99 1	98s 1	97s 0	96s 0	95s 0	94s 0	93s 0	92s 0
A8o 0	K8o 0	Q8o 0	J8o 0	T8o 0	98o 0	88 1	87s 0	86s 0	85s 0	84s 0	83s 0	82s 0
A7o 0	K7o 0	Q7o 0	J7o 0	T7o 0	97o 0	87o 0	77 1	76s 1	75s 0	74s 0	73s 0	72s 0
A6o 0	K6o 0	Q6o 0	J6o 0	T6o 0	96o 0	86o 0	76o 0	66 1	65s 1	64s 0	63s 0	62s 0
A5o 0	K5o 0	Q5o 0	J5o 0	T5o 0	95o 0	85o 0	75o 0	65o 0	55 1	54s 1	53s 0	52s 0
A4o 0	K4o 0	Q4o 0	J4o 0	T4o 0	94o 0	84o 0	74o 0	64o 0	54o 0	44 0	43s 0	42s 0
A3o 0	K3o 0	Q3o 0	J3o 0	T3o 0	93o 0	83o 0	73o 0	63o 0	53o 0	43o 0	33 0	32s 0
A2o 0	K2o 0	Q2o 0	J2o 0	T2o 0	92o 0	82o 0	72o 0	62o 0	52o 0	42o 0	32o 0	22 0

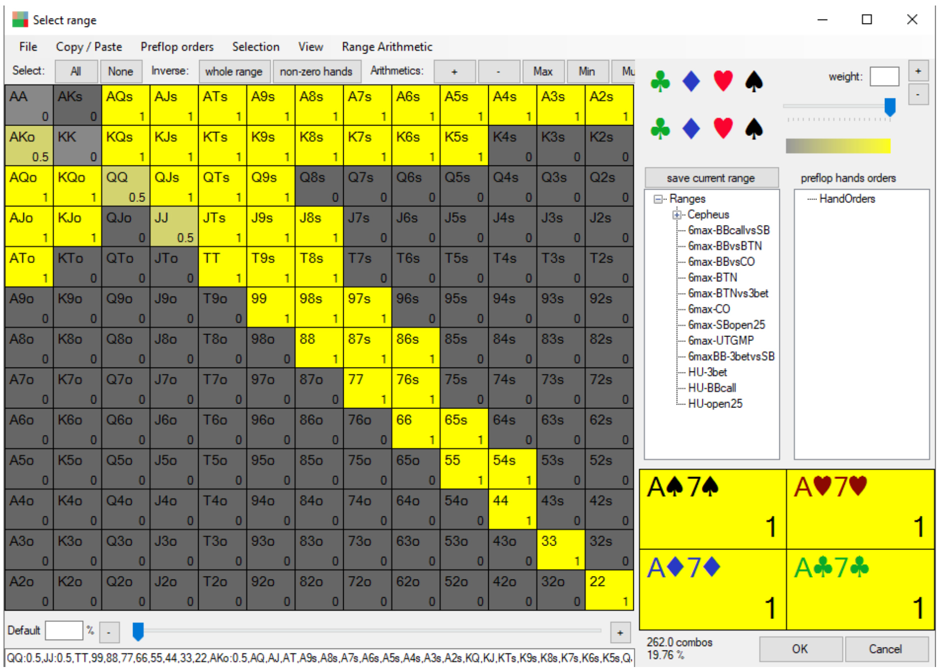
A♠J♠ 1	A♥J♥ 1
A♦J♦ 1	A♣J♣ 1

AA,KK,QQ,JJ,TT,99,88,77,66,55,AK,AQ,AJ,ATs,A9s,A8s,A7s,A6s,A5s,A4s,A3s,A2s,KQ,KJs,KTs,K9s,K8s,QJs,QTs,Q9s,JTs,J9s,T9s,T

220.0 combos
16.59 %

Figure 9 - PioSOLVER's range assigning tool (1)

We consider the player in the player in the small-blind, in this case referred to as "OOP" for "out-of-position", to raising every suited hand that contains an ace, every pair larger than 55, every broadway apart from QJo and KJo, as well as some suited connectors, namely 54s, 65s, 76s, 87s, 98s, T9s, T8s, J9s, K8s and K9s. The selected range consists of 220 hands, or 16.59% of all starting hands.



AA 0	AKs 0	AQs 1	AJs 1	ATs 1	A9s 1	A8s 1	A7s 1	A6s 1	A5s 1	A4s 1	A3s 1	A2s 1
AKo 0.5	KK 0	KQs 1	KJs 1	KTs 1	K9s 1	K8s 1	K7s 1	K6s 1	K5s 1	K4s 0	K3s 0	K2s 0
AQo 1	KQo 1	QQ 0.5	QJs 1	QTs 1	Q9s 1	Q8s 0	Q7s 0	Q6s 0	Q5s 0	Q4s 0	Q3s 0	Q2s 0
AJo 1	KJo 1	QJo 0	JJ 0.5	JTs 1	J9s 1	J8s 1	J7s 0	J6s 0	J5s 0	J4s 0	J3s 0	J2s 0
ATo 1	KTo 0	QTo 0	JTo 0	TT 1	T9s 1	T8s 1	T7s 0	T6s 0	T5s 0	T4s 0	T3s 0	T2s 0
A9o 0	K9o 0	Q9o 0	J9o 0	T9o 0	99 1	98s 1	97s 1	96s 0	95s 0	94s 0	93s 0	92s 0
A8o 0	K8o 0	Q8o 0	J8o 0	T8o 0	98o 0	88 1	87s 1	86s 1	85s 0	84s 0	83s 0	82s 0
A7o 0	K7o 0	Q7o 0	J7o 0	T7o 0	97o 0	87o 0	77 1	76s 1	75s 0	74s 0	73s 0	72s 0
A6o 0	K6o 0	Q6o 0	J6o 0	T6o 0	96o 0	86o 0	76o 0	66 1	65s 1	64s 0	63s 0	62s 0
A5o 0	K5o 0	Q5o 0	J5o 0	T5o 0	95o 0	85o 0	75o 0	65o 0	55 1	54s 1	53s 0	52s 0
A4o 0	K4o 0	Q4o 0	J4o 0	T4o 0	94o 0	84o 0	74o 0	64o 0	54o 0	44 1	43s 0	42s 0
A3o 0	K3o 0	Q3o 0	J3o 0	T3o 0	93o 0	83o 0	73o 0	63o 0	53o 0	43o 0	33 1	32s 0
A2o 0	K2o 0	Q2o 0	J2o 0	T2o 0	92o 0	82o 0	72o 0	62o 0	52o 0	42o 0	32o 0	22 1

A♠7♠ 1	A♥7♥ 1
A♦7♦ 1	A♣7♣ 1

QQ:0.5,JJ:0.5,TT,99,88,77,66,55,44,33,22,AKo:0.5,AQ,AJ,AT,A9s,A8s,A7s,A6s,A5s,A4s,A3s,A2s,KQ,KJ,KTs,K9s,K8s,K7s,K6s,K5s,Q

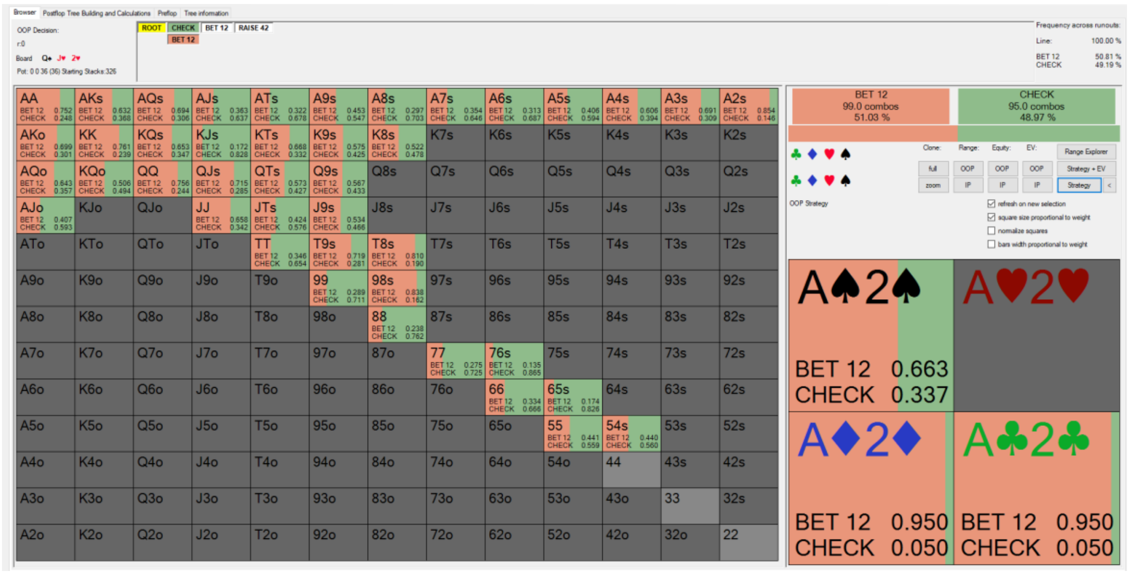
262.0 combos
19.76 %

Figure 10 - PioSOLVER's range assigning tool (2)

We consider the player in the button, shown as “IP” for “in-position”, to be calling 262 hands or 19.76% of all starting hands, as described in the figure. There are several important takeaways from the defined range. The fact that the strongest possible hands, AA, AKs and KK are not included in this player’s range is due to the fact that we do not consider this player to be ever calling with these hands, always playing this with a re-raise. The concept of hand weight is also present, with the player calling half of the times with the hands AKo, QQ and JJ, giving these a 0.5 weight in this range.

Having defined both player’s ranges and the pot details, which in this case consist of a starting pot value of 36 chips and an effective stack size of 325 chips, it is necessary to input the allowed actions and bet sizes for both players in the flop, turn and river. We allow the in-position player to bet 33.55% of the pot on the flop, and 66% on the turn and river. We also allow him to raise to three times the size of the pot on the flop, as well as 50% of the pot on the turn and river, as well as move all-in. The out-of-position player configuration is similar to that of the in-position player’s in every parameter except for the flop raise size, which we define as 50%.

With every arbitrary parameter defined, the software is ready to build the game tree based on which it will simulate the game in order to derive the optimal solution. While it is common for the degree of imperfection associated with the solutions to be measured in Epsilon-equilibrium, PioSOLVER presents it as “exploitability per hand”, representing the chip amount a perfect adversary – one knowing our exact strategy and employing the best possible counter strategy – would expect to win per hand (*Technical Details - PioSOLVER*, 2015). In order to obtain a reasonable approximation from which we can derive useful information, it is advisable to solve until the solution presents an exploitability of 0.25% to 1% of the value of the pot.



OOP Decision:
r:0
Board Q♠ J♥ 2♥
Pot: 0 0 36 (36) Starting Stacks:326

Frequency across runouts:
Line: 100.00 %
BET 12 50.81 %
CHECK 49.19 %

AA BET 12 0.752 CHECK 0.248	AKs BET 12 0.632 CHECK 0.368	AQs BET 12 0.694 CHECK 0.306	AJs BET 12 0.363 CHECK 0.637	ATs BET 12 0.322 CHECK 0.678	A9s BET 12 0.453 CHECK 0.547	A8s BET 12 0.297 CHECK 0.703	A7s BET 12 0.354 CHECK 0.646	A6s BET 12 0.313 CHECK 0.687	A5s BET 12 0.406 CHECK 0.594	A4s BET 12 0.606 CHECK 0.394	A3s BET 12 0.691 CHECK 0.309	A2s BET 12 0.854 CHECK 0.146
AKo BET 12 0.699 CHECK 0.301	KK BET 12 0.761 CHECK 0.239	KQs BET 12 0.653 CHECK 0.347	KJs BET 12 0.172 CHECK 0.828	KTs BET 12 0.668 CHECK 0.332	K9s BET 12 0.575 CHECK 0.425	K8s BET 12 0.522 CHECK 0.478	K7s	K6s	K5s	K4s	K3s	K2s
AQo BET 12 0.643 CHECK 0.357	KQo BET 12 0.506 CHECK 0.494	QQ BET 12 0.756 CHECK 0.244	QJs BET 12 0.715 CHECK 0.285	QTs BET 12 0.573 CHECK 0.427	Q9s BET 12 0.567 CHECK 0.433	Q8s	Q7s	Q6s	Q5s	Q4s	Q3s	Q2s
AJo BET 12 0.407 CHECK 0.593	KJo	QJo	JJ BET 12 0.658 CHECK 0.342	JTs BET 12 0.424 CHECK 0.576	J9s BET 12 0.534 CHECK 0.466	J8s	J7s	J6s	J5s	J4s	J3s	J2s
ATo	KTo	QTo	JTo	TT BET 12 0.346 CHECK 0.654	T9s BET 12 0.719 CHECK 0.281	T8s BET 12 0.810 CHECK 0.190	T7s	T6s	T5s	T4s	T3s	T2s
A9o	K9o	Q9o	J9o	T9o	99 BET 12 0.289 CHECK 0.711	98s BET 12 0.838 CHECK 0.162	97s	96s	95s	94s	93s	92s
A8o	K8o	Q8o	J8o	T8o	98o	88 BET 12 0.238 CHECK 0.762	87s	86s	85s	84s	83s	82s
A7o	K7o	Q7o	J7o	T7o	97o	87o	77 BET 12 0.275 CHECK 0.725	76s BET 12 0.135 CHECK 0.865	75s	74s	73s	72s
A6o	K6o	Q6o	J6o	T6o	96o	86o	76o	66 BET 12 0.334 CHECK 0.666	65s BET 12 0.174 CHECK 0.826	64s	63s	62s
A5o	K5o	Q5o	J5o	T5o	95o	85o	75o	65o	55 BET 12 0.441 CHECK 0.559	54s BET 12 0.440 CHECK 0.560	53s	52s
A4o	K4o	Q4o	J4o	T4o	94o	84o	74o	64o	54o	44	43s	42s
A3o	K3o	Q3o	J3o	T3o	93o	83o	73o	63o	53o	43o	33	32s
A2o	K2o	Q2o	J2o	T2o	92o	82o	72o	62o	52o	42o	32o	22

BET 12	CHECK
99.0 combos	95.0 combos
51.03 %	48.97 %

♣ ♦ ♥ ♠
♣ ♦ ♥ ♠
OOP Strategy

☑ refresh on new selection
☑ square size proportional to weight
☐ normalize squares
☐ bars width proportional to weight

A♠2♠ BET 12 0.663 CHECK 0.337	A♥2♥
A♦2♦ BET 12 0.950 CHECK 0.050	A♣2♣ BET 12 0.950 CHECK 0.050

Figure 11 - PioSOLVER example output

Once calculated, the solution is browsable on all nodes of the game tree, presenting the optimal frequency the player in a decision point should perform each action with each his hole cards. Each color signifies an action, with red representing bet, green representing check, and gray representing the cards that are not in the player's defined range. The proportion of color filling the background of each holding is relative to the frequency of the action. Analyzing Figure 11 - PioSOLVER example output, we can see that the optimal betting frequency for the given situation is 51.03%, with about 99 hand combinations betting. Because of the proximity between the betting and checking frequencies, acquiring this information in this particular situation wouldn't be of much use if the frequencies by which optimal play is found weren't available for each of the individual holdings in the player's range. These are also analyzable – the scope can be narrowed to individual hands, demonstrating, for example, that the out-of-position player should raise AA with about 80% frequency, KQs with about 50% frequency, and TT with about 33% frequency. The exact values for each action are displayed and hovering the mouse over a certain holding will display additional information, such as the fact that the out-of-position player should always raise A2s of clubs and diamonds, but only raise the spades suited hand two-thirds of the time.

Elations that can be obtained from the provided solution are numerous. However, simplifications are necessary in order for a human player to follow the provided strategy – while it is feasible for a human player to integrate always raising his lowest and highest suited ace hands while checking AJs to A5s half of the time into his strategy, it is unrealistic to consider a player can apply the exact action frequency obtained by the solver in the most various game situations, considering that on later streets of the game the solver can apply different frequencies for multiple raise sizes as well as a check action on each hand.

While this software offers additional analysis features that are not going to be described in this study, such as equity and expected value explorers, it is relevant to mention that a pre-flop solver was made available 9 months after it's market debut (*The preflop solver - PioSOLVER*, 2015). Due to the RAM and disk space requirements associated with solving a full game with the pre-flop, flop, turn and river stages, an arbitrary subset of flops is necessary to be input in order to perform the solve. This is a necessary abstraction in order to enable users with ordinary computers to use this feature, and as demonstrated on chapter 2.3.2, it should be noted that selecting a subset of flops that adheres to certain criteria can create a solution that is approximate to that achieved by solving for the entirety of the possible flops. Combining this abstraction with the restriction of stack and bet sizes, a tree describing a heads-up game when the in-position player can open fold, limp or raise can be simplified to the format that follows.

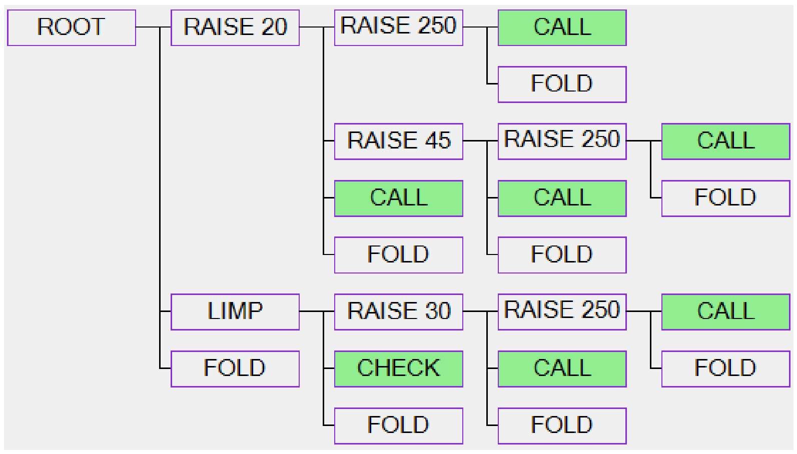


Figure 12 - PioSOLVER's example preflop game tree

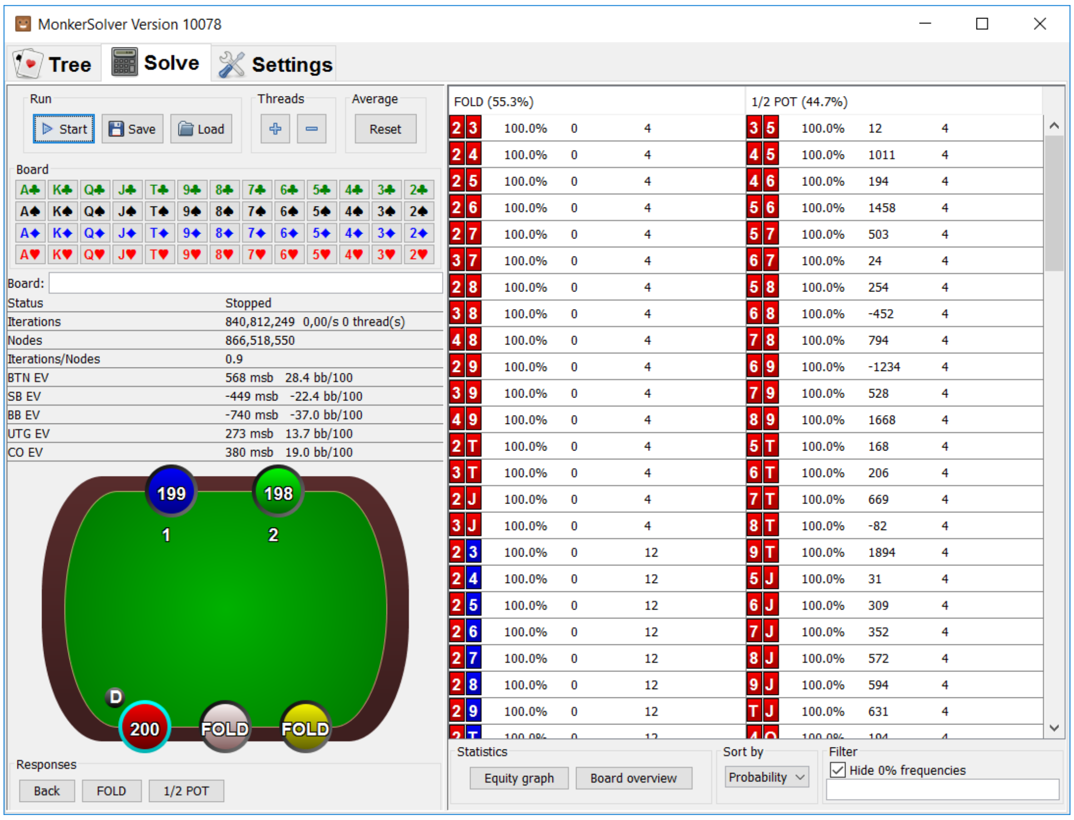
The ability to provide human-interpretable solution trees and the relatively low processor capacity requirements were breakthrough factors for the widespread adoption of this software in the poker community, being equally as relevant as the game solving approach it employed. A whole new market was created for this type of software, with a large portion of professional poker players taking advantage of its capacities by utilizing it in order to deterministically study game situations, when the previously available alternative was to strategically discuss it with others.

Several alternatives to PioSOLVER have been made available in the market during the upcoming years, such as MonkerSolver (*MonkerWare.com - MonkerSolver*, 2020), SimplePostFlop (*SimplePostFlop*, no date), GTO+ (*GTO+ – Making Game Theory Practical*, no date) and jesolver (*jeskola.net/jesolver_beta/*, no date), some of which will be discussed in the following chapters.

3.2.2 MonkerSolver

Launched in February 2017, MonkerSolver (*MonkerWare.com - MonkerSolver*, 2020) quickly cemented its market position due to its ability to solve Omaha, a poker variant that plays similarly to Hold’em in most aspects, with the main distinction between them residing in the player’s starting cards. In this variant, players’ starting hands contain four cards, from which they must use exactly two in order to make a hand, meaning that while in Hold’em a player can make a flush with one of his hole cards if there are four cards of the same suit on the board, this is not the case in Omaha(Chad Holloway, 2017). Additional distinctions between these variants are not a factor of the game’s rules but rather the dynamics these create. It is easily deductible that the game is more prone to action with players being two times as likely to attain a made hand or draw. Relative hand strength is more difficult to ascertain – while the best ranked hand in Hold’em (Ax Ax) is an 83% favourite versus the second best ranked hand (Kx Kx), in Omaha the best ranked hand (Ax Ax Kh Ks) has only a

31% chance of winning, and 41% change of tying, against the second best ranked hand (Ax Ax Td Jc). Therefore, and likely in part due to the aforementioned game dynamics, the most popular Omaha betting structure is pot-limit, which as previously demonstrated reduces a solution's number of nodes.



Status	Stopped
Iterations	840,812,249 0,00/s 0 thread(s)
Nodes	866,518,550
Iterations/Nodes	0.9
BTN EV	568 msb 28.4 bb/100
SB EV	-449 msb -22.4 bb/100
BB EV	-740 msb -37.0 bb/100
UTG EV	273 msb 13.7 bb/100
CO EV	380 msb 19.0 bb/100

FOLD (55.3%)				1/2 POT (44.7%)			
2 3	100.0%	0	4	3 5	100.0%	12	4
2 4	100.0%	0	4	4 5	100.0%	1011	4
2 5	100.0%	0	4	4 6	100.0%	194	4
2 6	100.0%	0	4	5 6	100.0%	1458	4
2 7	100.0%	0	4	5 7	100.0%	503	4
3 7	100.0%	0	4	6 7	100.0%	24	4
2 8	100.0%	0	4	5 8	100.0%	254	4
3 8	100.0%	0	4	6 8	100.0%	-452	4
4 8	100.0%	0	4	7 8	100.0%	794	4
2 9	100.0%	0	4	6 9	100.0%	-1234	4
3 9	100.0%	0	4	7 9	100.0%	528	4
4 9	100.0%	0	4	8 9	100.0%	1668	4
2 T	100.0%	0	4	5 T	100.0%	168	4
3 T	100.0%	0	4	6 T	100.0%	206	4
2 J	100.0%	0	4	7 T	100.0%	669	4
3 J	100.0%	0	4	8 T	100.0%	-82	4
2 3	100.0%	0	12	9 T	100.0%	1894	4
2 4	100.0%	0	12	5 J	100.0%	31	4
2 5	100.0%	0	12	6 J	100.0%	309	4
2 6	100.0%	0	12	7 J	100.0%	352	4
2 7	100.0%	0	12	8 J	100.0%	572	4
2 8	100.0%	0	12	9 J	100.0%	594	4
2 9	100.0%	0	12	T J	100.0%	631	4

Figure 13 - MonkerSolver's user interface

A greater novelty than Omaha solving was the fact that this software is capable of generating solutions for sub-games with more than two players on all betting streets, including the pre-flop. While theoretically unsound as previously demonstrated in chapter 2.3.3, multiway sub-game solving became a highly sought-after feature, in particular by players of Sit & Go type games. Certain characteristics of this type of games makes the application of multi-way solving more feasible – multiple players to reach the flop with a lower frequency, the actions in the preflop street gain importance due to the shorter stacks and blind pressure, and players tend to adhere to a defined strategy more constantly than in other formats – but they do not guarantee the player following an equilibrium strategy not to lose, unlike what occurs in heads-up scenarios.

Revealing no information in regard to its multi-way solving methodology, MonkerSolver does claim to use "state of the art techniques to reduce game size"(*MonkerWare.com - MonkerSolver*, 2020) in the form of a technique referred to as "bucketing"(*MonkerWare.com - Abstraction*, 2020). This technique consists in grouping strategically similar hands in order to reduce the size of the game tree in a considerable manner while maintaining the significance of the solution. The software resorts to this type of abstraction in two particular situations. The first use occurs when calculating preflop situations in the game of Hold'em, and follows the principles described in chapter 2.3.2, with different levels of abstraction being compared to the University of Alberta's full game solution (Tammelin *et al.*, 2015) so as to provide an indication of how the reduction of the game size affects the preflop solution (*MonkerWare.com - Abstraction*, 2020). Such comparison can be viewed in the figure that follows.

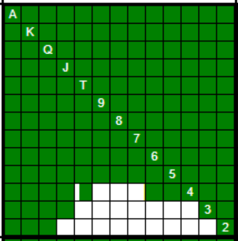
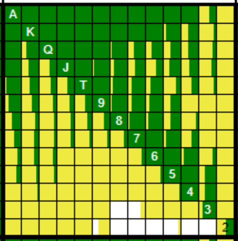
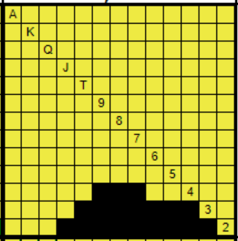
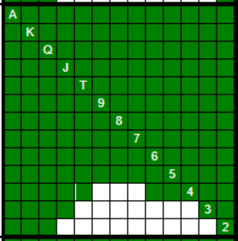
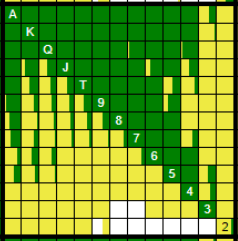
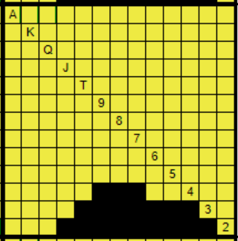
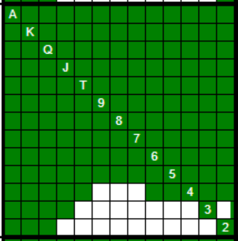
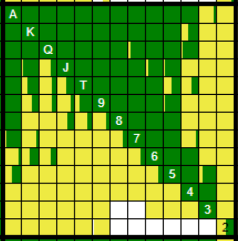
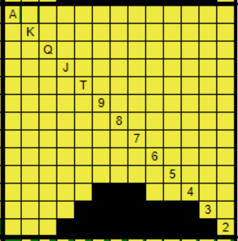
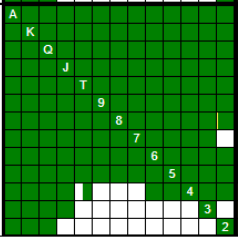
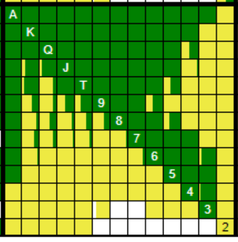
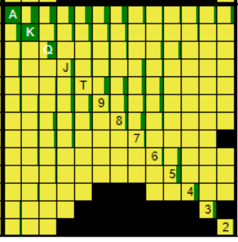
Abstraction	No actions	Bet	Bet, Raise	Game size	Mistakes
Cepheus (Full game)				1	-
30/Large				$^{1}/_{19,000}$	No clear mistakes.
30/Small				$^{1}/_{395,000}$	Does not always open 32s.
30/None				$^{1}/_{1,150,000}$	Does not open 32s or 72s. Sometimes folds 93o vs open. Has a 4bet range.

Figure 14 - Solution quality comparison between multiple abstraction techniques

The second use occurs when calculating Omaha situations, a game for which the developers of this software claim that even a post-flop only solution is unfeasible to be generated without the use of heavy abstractions. While the principle is maintained – grouping strategically similar hands – it is in this case applied by merging cards according to their potential of reaching a flush. Before the river, only suits capable of becoming flushes are considered, while only suits with three or more cards on the board are considered on the river. This type of abstraction is referred to as "texture abstraction".

One shortcoming in this software is the lack of an exploitability measure associated with the presented solution, which hinders their evaluation on a theoretical level. Despite this, tests performed by users on a benchmark blind-versus-blind simulation showed that the solution presented by MonkerSolver can closely relate to that output by PioSOLVER with abstraction settings set to one of the following:

- 90 buckets, no abstraction on the flop (Full game), small abstraction on the turn and river (30/Large);
- 30 buckets, no abstraction on the flop and turn, minor abstraction on the river (30/Large).

Despite the efforts put into reducing the game size without creating a degree of noise that would degrade the quality of the solution to a point where it had major flaws, simulations for Omaha games and multi-way scenarios require enormous amounts of RAM, which essentially made these features only available to those with the resources to rent or acquire server-grade computers.

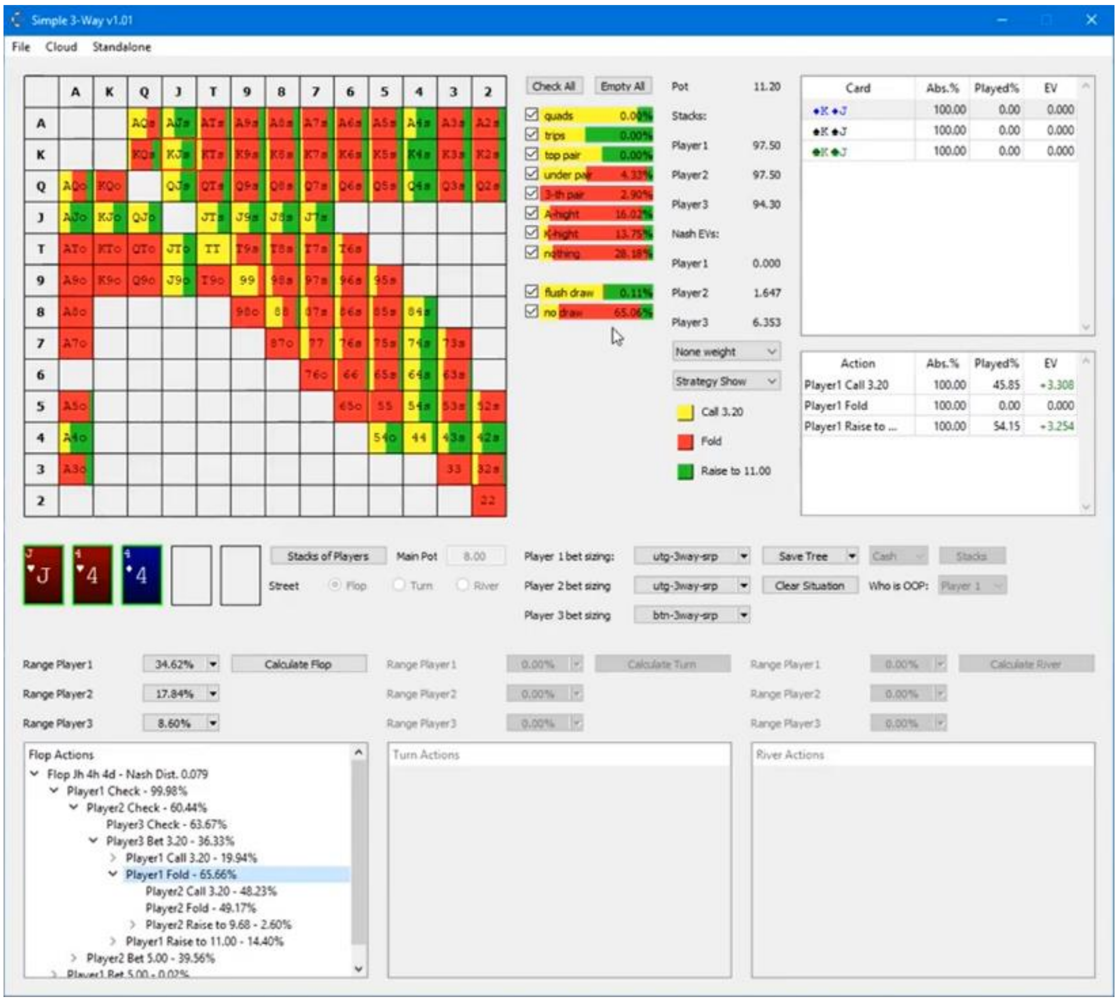
3.2.3 SimplePostFlop

While largely similar to the previously mentioned solver softwares, SimplePostFlop (*SimplePostFlop*, no date) differed in the fact that it offered various calculation algorithms for the user to choose from, doing so in order to enable users equipped with less capable computers to run Poker simulations.

Available algorithms are classified in 2 different categories, the first being vector algorithms – Local A1, Local A2 and Local A3 – and the remaining being Monte Carlo simulation based algorithms – ESCFR and CSCFR. The algorithms positioned in the last category are only available for usage on the flop, with the turn and river calculation options consisting solely of the vector algorithms. Analyzing the same situation using different algorithm combinations for the multiple betting streets can provide insight on how the quality of the solutions achieved relate with the time elapsed and memory required to calculate it. The different algorithms generally vary on the following characteristics:

- **Convergence –** Some algorithms are not guaranteed to converge, and some take more time to do so than their alternatives;
- **Precision –** The minimum nash distance achieved in the solution is different between algorithms and their combinations;
- **Memory requirements –** Some algorithms can require twice as much RAM compared to their alternatives.

In terms of abstractions, the software can calculate using 4 different settings – Small, Medium, Large and Perfect – which in similarity with MonkerSolver's settings (see Figure 14 - Solution quality comparison between multiple abstraction techniques) to the completeness of the game tree in which the game will be simulated, in what is assumed to resemble the methodologies described in 2.3.2. An approximation mode can also be defined in which the selected first percentage of the calculations are approximated, and the remaining calculated to exactitude.



quads	0.00%
trips	0.00%
top pair	0.00%
under pair	4.33%
3-th pair	2.90%
A-hight	16.02%
K-hight	13.75%
nothing	28.18%
flush draw	0.11%
no draw	65.06%

Pot	11.20
Stacks:	
Player1	97.50
Player2	97.50
Player3	94.30
Nash EVs:	
Player1	0.000
Player2	1.647
Player3	6.353

None weight
Strategy Show

🟨 Call 3.20
🟥 Fold
🟩 Raise to 11.00

Card	Abs.%	Played%	EV
♦K ♦J	100.00	0.00	0.000
♠K ♠J	100.00	0.00	0.000
♣K ♣J	100.00	0.00	0.000

Action	Abs.%	Played%	EV
Player1 Call 3.20	100.00	45.85	+3.308
Player1 Fold	100.00	0.00	0.000
Player1 Raise to ...	100.00	54.15	+3.254

Main Pot 8.00
Street [✓ Flop] Turn River

Player 1 bet sizing:	utg-3way-srp
Player 2 bet sizing	utg-3way-srp
Player 3 bet sizing	btn-3way-srp
Who is OOP:	Player 1

Range Player1	34.62%	Range Player1	0.00%	Range Player1	0.00%
Range Player2	17.84%	Range Player2	0.00%	Range Player2	0.00%
Range Player3	8.60%	Range Player3	0.00%	Range Player3	0.00%

```
Flop Actions
Flop Jh 4h 4d - Nash Dist. 0.079
  Player1 Check - 99.98%
    Player2 Check - 60.44%
      Player3 Check - 63.67%
      Player3 Bet 3.20 - 36.33%
        Player1 Call 3.20 - 19.94%
        Player1 Fold - 65.66%
          Player2 Call 3.20 - 48.23%
          Player2 Fold - 49.17%
          Player2 Raise to 9.68 - 2.60%
        Player1 Raise to 11.00 - 14.40%
    Player2 Bet 5.00 - 39.56%
  Player1 Bet 5.00 - 0.02%
```

Figure 15 - SimplePostFlop's user interface

Results are displayed in an interface similar to that of PioSOLVER's, with the optimal action frequency for each individual hand, and the equity and expected value figures for all players analyzed.

It is important to note that SimplePostFlop is only part of the software package made available by Simple Poker(Poker, no date b), bundling with Simple Preflop Hold'em with the capability of solving preflop situations, Simple 3-way (Poker, no date a), enabling the calculation of equilibrium strategies for 3 player situations and Simple Omaha, which is to the Omaha variant what SimplePostFlop is to Hold’em.

3.3 Overview

The continuous evolution of academic poker bots and solvers demonstrated in this chapter shows that there is an ever growing amount of resources being made available which use machine learning methodologies in order to allow humans to grasp aspects of the game which were previously unknown.

With the continuous increase in computational performance accompanied by a decrease in the price of components, it is foreseen that the game of No Limit Texas Hold’em will inevitably reach a state where it is considered effectively solved, just as it’s Limit Texas Hold’em counterpart, and it’s Chess and Checkers ancestors.

4 Value analysis

This chapter provides an in-depth view of the Value Analysis (VA) process applied to the solution in development with the main objective of identifying the optimal ratio of production cost to value offered. This provides a measure of how the requirements and overall functionality package affects the development process in terms of effort and time-to-market, which translates into the products market viability (Rich and Holweg, 2000).

A Value Engineering (VE) approach was adopted for conducting the aforementioned process, applying many of the same principles and techniques to the manufacturing stages.

A product or service's value is strongly related to a set of attributes (Rich and Holweg, 2000), some of which relate to the product's functionality and others to the product's value from the consumer point of view:

- **Utility value:** how functional a product or service intends to be;
- **Esteem value:** the value a user or customer attributes to the product or service's attributes, not directly contributing to utility but aesthetic and subjective value instead;
- **Market value:** what is the market prepared to pay for the product or service's usage, consisting of the sum between the aforementioned attributes ($Market\ value = Utility\ value + Esteem\ value$).

The Value Analysis and Value Engineering processes are systematical, formal and organized methodologies which can be described by the following figure.

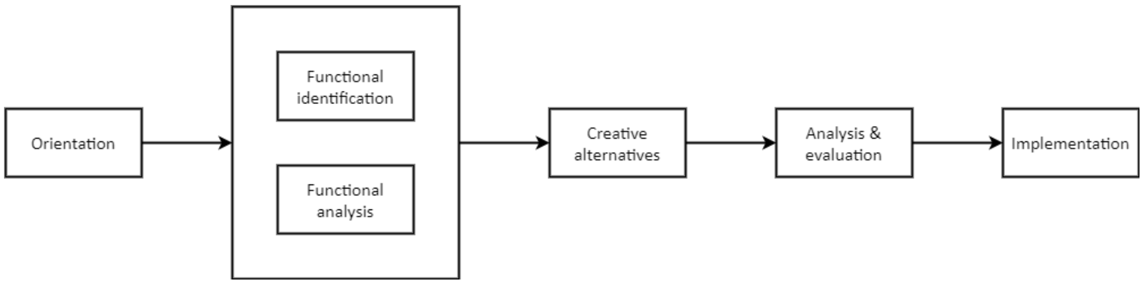


Figure 16 – The Value Analysis process adapted from Rich and Holweg, 2000; Nicola, 2019

In order to correctly assess and direct a products function in order to meet the market's demand, a profound understanding between the product's functional specification, how it provides an answer to a market's inefficiency and how it's value will ultimately be perceived by the end customer. The phases that compose the Value analysis process (Figure 16) concede this assessment, and can be described as follows (Rich and Holweg, 2000):

- **Orientation:** selection, preparation and presentation of the product to be adopted during the Value Analysis process;
- **Functional identification and analysis:** analysis of the product and identification of it's primary functionality, considering the market and consumers;
- **Creative alternatives:** identifying solutions that enable the achievement of the product's intended functions;
- **Analysis & evaluation:** evaluating the cost-benefit relation in each of the solutions derived in the previous phase in order to identify which is more adequate to the project;
- **Implementation:** briefly described the solution to be implemented based on the conclusions reached through this procedure.

4.1.1 Orientation

The orientation phase consists of the selection, preparation and presentation of the product as well as the creation of the value analysis team.

In this stage of the analysis, the New Concept Development (NCD) was adopted in order to present the product's environment, providing a common language representation and definition of a product's innovation process (Koen *et al.*, 2002).

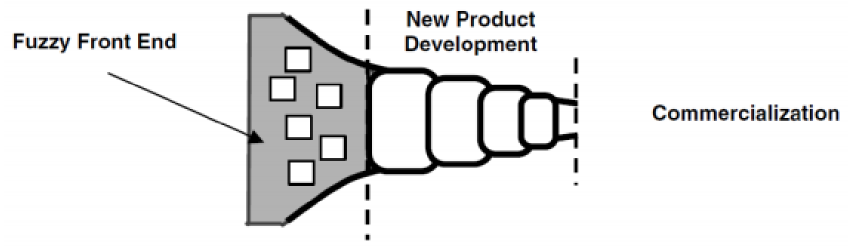


Figure 17 – Product innovation process adapted from (Koen *et al.*, 2002; Nicola, 2019)

This innovation process is composed by 3 phases, as described by Figure 17:

1. Fuzzy Front End (FFE), composed by unpredictable, unorganized and unstructured activities;
2. New Product Development (NPD), composed by structured activities;
3. Commercialization of the resulting product.

The NCD model adapts the activities to be developed in the FFE specifying no particular order, enabling their execution when deemed appropriate. It is a relationship model composed of the engine – senior and executive-level support – and the elements it powers, as well as influencing factors.

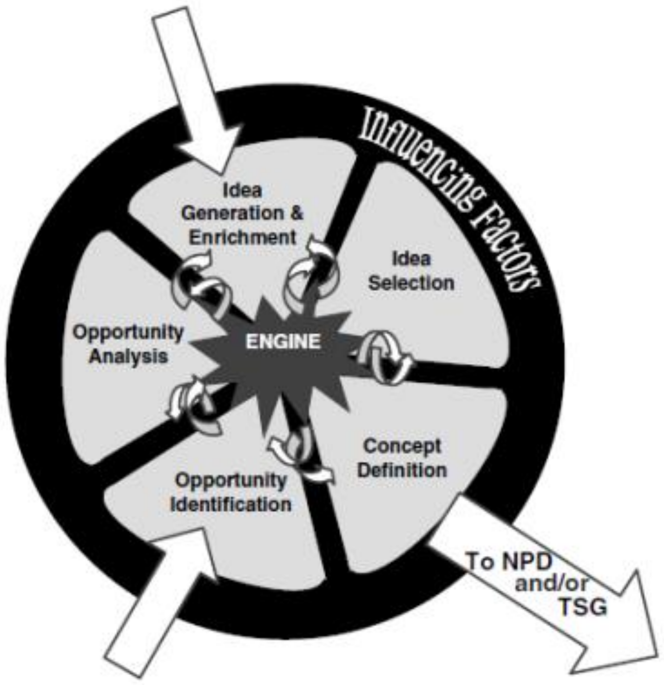


Figure 18 – New concept development model adapted from (Koen *et al.*, 2002; Nicola, 2019)

4.1.1.1 Influencing factors

These are factors derived from the product or service provider's environment and consist of the organizational capabilities, customer, competitor and outside world influences and capabilities provided by the scientific or technological resources in use (Koen *et al.*, 2002). In their whole, these dictate the provider's business strategies and competitive factors.

The influencing factors identified for this study's project are the following:

1. Technological adequacy and capability;
2. Maturity and complexity of the concepts to apply;
3. Regulatory and legal influences;
4. Customer influences;
5. Organizational capabilities.

With chapter 2.3.1 in mind, it is immediately evident that the processing and storage capabilities required to generate, store and access a full No Limit Hold'em solution tree in real-time are yet to be available even to super computers as of the date of this study. Moore's law (Moore, 2004) can be utilized in order to estimate when such capacities might be available, but in order to compete with the solution's remaining market players, the abstraction techniques regarded in chapter 2.3.2 have to be considered. More so, the solution to implement must consider how a technology might adapt to the specific project's requirements, due to the task having very specific requirements which are not typically considered by software products.

The academic Poker bot market described in chapter 3.1 shows consistent, year-over-year innovation in the machine learning algorithms applied, as well as improvements to the previous state-of-the-art contenders, which not only consolidates the maximum viable time for the development process of a product which intends to present disruptive capabilities, but also allows for a much smaller time-frame for researchers and engineers to grasp the methodologies applied and establish how to improve based on them.

Regulatory, legal and customer entities influence the development process due to their capacity of endorsing Poker as the study-object of scientific research in the field of artificial intelligence, or altogether deeming the creation of autonomous playing agents as a frowned-upon practice. Online Poker is still ongoing its regulatory transition in a large percentage of the world's countries (*Europe Poker Laws - Best Poker Sites in Europe Today*, no date), and online Poker networks consistently battle against bots fearing they lead to the alienation of their player base (*partypoker closes a further 121 bot accounts in July 2019; $76,267 and €88,351 seized*, 2019).

Finally, and in relation to the organizational capabilities which "determine whether and how opportunities are identified and analyzed, how ideas are selected and generated, and how concepts and technologies are developed" (Koen *et al.*, 2002), one must consider that some of state-of-the-art presented in chapter 3.1 is a result of decades of work by teams of researchers equipped with enormous computing power, and adapt expectations regarding the final product.

4.1.1.2 Engine

The NCD model is based on leadership, corporate culture and business strategy, the main drivers for a company's innovation process (Koen *et al.*, 2002). The application of appropriate methodologies regarding these three corporate facets assure value generation through a combination of new products and services.

For the purposes of this study, these do not entirely constitute what can be defined as the engine, but it can be stated that the objective it is intended to accomplish is that of studying and analyzing a selection of techniques from the wide array of existing ones which will ultimately lead to the implementation of a system that is able to relate with the previously existing ones in certain aspects, most notably in terms of performance. This factor, in combination with the ambition to grasp how these concepts may solve real-life problems outside of this study's target object – Poker – position this product as a pilot showcase of the capabilities of the machine learning capabilities described in chapter 2.4.

4.1.1.3 Opportunity identification

An opportunity can be defined as a business or technological gap between the current solutions and the ones envisioned which can be taken advantage of by a company or individual in order to gain a competitive advantage, respond to a threat, solve a problem or ameliorate a difficulty (Nicola, 2019).

This study's opportunity lies in the currently observable use-cases in the market for the application of algorithms similar to the one described in the specification of this study's solution, as well as the exploration of the scientific advancements that the employment of machine learning techniques might have in the most varied fields of study. While the latter has a strictly academical purpose, the use-cases for such an algorithm are already present, and not only in the market of the game of Poker, although that is the one on focus in this study.

The most immediate use-cases are the following:

- Creation of analysis software such as that described in chapter 3.2;
- Creation of gaming applications which providing the player with the aftermath of each decision, serving an educational purpose while enabling the improvement of their game – an example of this is PokerSnowie[19];
- Commercializing a SaaS[20] solution which empowers the addition of features such as the one described in the previous use-case to Poker networks' software, enabling them to provide players with insights while playing a real game. This is important for such networks from a marketing and customer retention standpoint, and an example of such a solution can already be seen in production in the PartyPoker network's MyGame functionality[21] or as a SaaS package[22];
- Developing pattern analysis techniques that aid in the identification of other autonomous agents through their in-game tendencies.

4.1.1.4 Opportunity analysis

Having identified the valid opportunities from a business perspective, the necessity arises to analyze them separately, assessing their capability of providing value to the end user.

It is important to realize how the game of Poker has evolved since the Moneymaker effect (Swains, 2006) catapulted it into the spotlight in the year of 2003. With a massive rise in popularity and an increase in accessibility provided by the recently founded online Poker rooms such as PokerStars, founded in 2001, and Full Tilt Poker, founded in 2004, the game of Poker experienced an unprecedented surge of new players which became interested in the game due to the perceived potential of obtaining exponential profits, along with the association of a certain degree of skill being required in order to correctly play the game. The rise of Internet accessibility and TV coverage of the World Series of Poker events propagated the game throughout varied demographics, and the number of players grew for several years, reaching its pinnacle somewhere around April 15th, 2011 – Black Friday (Popper, 2011). This was when the American Department of Justice accused PokerStars, Full Tilt Poker and the Cereus network, which fathered other Poker sites, with several fraud related charges, effectively paralyzing the entire operation and leaving the account balances of millions of players inaccessible. This event is perceived as a major blow to the

[19] https://www.pokersnowie.com/
[20] SaaS stands for "Software as a service"
[21] https://www.partypoker.com/en/how-to-play/my-game-explained
[22] https://neopokerlab.com/white-label-poker

economy of online Poker, with state efforts to regulate and tax the game subsequent to this date being one of the commonly attributed causes for the game’s decline.

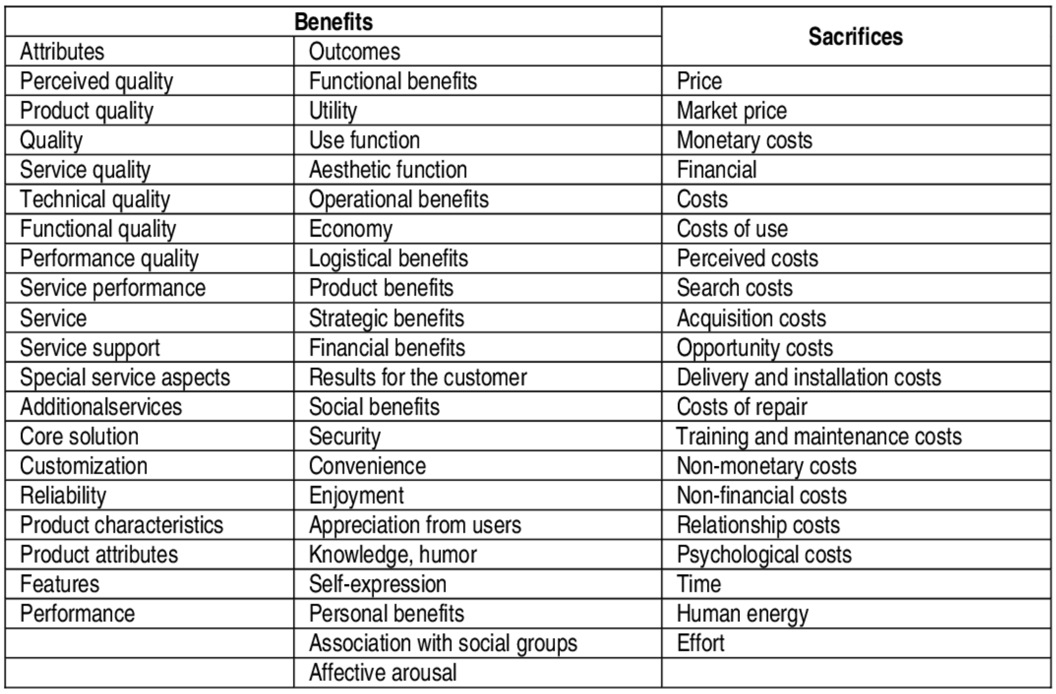
With less novice players entering the market and many of the previous staying – some of which were professional electronic sports competitors in other games – the game of Poker became increasingly hard due to the study efforts continuously put into it over the years, as well as coaching material, and most recently, simulation software, which allowed players to grasp aspects of optimal theories which lead to the application of near-unexploitable strategies which demonstrate elevated profitability, but more importantly, the creation of bots.

Bots are particularly nefarious in online Poker, not only because of their playing tendencies, which can as of the date of this study far surpass that of any human even with a simplified solution, but also because there have been in-the-wild examples of how bot rings might be created in order to collude by sharing their hole cards and soft-playing each other, massively exploiting unsuspecting players.

As more and more cases were uncovered, regular players became aware of how the game might be unfair and therefore became untrustworthy of the online Poker networks, which in turn launched anti-bot campaigns in which they demonstrate their efforts in eliminating non-human players from their tables, with this factor going as far as being presented as a competitive advantage in relation to the remaining networks.

Based on the aforementioned information and regarding the end-user standpoint in this matter, an evaluation of the opportunity and it’s benefits and sacrifices can be obtained. In the case of this study, the most interested parties are the player, benefitting from training, advice or protection, and the Poker networks, benefitting from offering the aforementioned qualities to their prospective users.

Conducting this analysis from a user’s standpoint is of major importance to analyze the opportunity’s value, since the party responsible for the project’s development usually holds a value perception that differs from that of the end-user (Nicola, 2019). To respond to this necessity, Woodall (2003) created the table shown in Figure 19 – Benefits and sacrifices table, which provides a value to cost ratio and allows for the assessment of an opportunity’s viability.



Benefits		Sacrifices
Attributes	Outcomes	
Perceived quality	Functional benefits	Price
Product quality	Utility	Market price
Quality	Use function	Monetary costs
Service quality	Aesthetic function	Financial
Technical quality	Operational benefits	Costs
Functional quality	Economy	Costs of use
Performance quality	Logistical benefits	Perceived costs
Service performance	Product benefits	Search costs
Service	Strategic benefits	Acquisition costs
Service support	Financial benefits	Opportunity costs
Special service aspects	Results for the customer	Delivery and installation costs
Additionalservices	Social benefits	Costs of repair
Core solution	Security	Training and maintenance costs
Customization	Convenience	Non-monetary costs
Reliability	Enjoyment	Non-financial costs
Product characteristics	Appreciation from users	Relationship costs
Product attributes	Knowledge, humor	Psychological costs
Features	Self-expression	Time
Performance	Personal benefits	Human energy
	Association with social groups	Effort
	Affective arousal	

Figure 19 – Benefits and sacrifices table (Woodall, 2003)

In order to conduct such an assessment, the benefits and sacrifices are identified for both the players and the service providers (Poker networks) from product, service and relationship standpoints.

The following table demonstrates the benefits to sacrifices relation the service providers are expected to experience.

Table 3 – Perceived value for service providers

Domain \ Scope	Product	Service	Relationship
Benefits	• Quality enhancement; • Additional features provided in comparison to competitors;	• Additional services provided in comparison to competitors; • Strategic benefits in the form of early-bird advantage;	• Enhance the product's perceived quality; • Financial benefits in terms of customer's results; • Gather enjoyment and appreciation from users
Sacrifices	• Time, human energy and effort resources spent on development; • Maintenance of the interface;	• Search or acquisition costs; • Time, human energy and effort spend on maintenance;	• Training and maintaining a team to manage this feature;

In turn, Table 4 demonstrates the same relation for the end-user, the player who gets to experience the possible additional features provided by the opportunities being studied.

Table 4 – Perceived value for users

Domain \ Scope	Product	Service	Relationship
Benefits	• In-game performance improvement; • Special service aspects in the format of tailored recommendations;	• Enhanced perception of reliability;	• Enhance the product's perceived quality; • Increase in client satisfaction;
Sacrifices	• Possibility of increased costs in the format of rake;	• Initial adaptation to changes or additions;	• Initial training and adaptation to the new features introduced;

The aforementioned benefit to sacrifices ratio remain true for both the service providers and consumers regardless of the opportunity in study, from the 4 hypothesis presented in the opportunity identification chapter (4.1.1.3), as all uniformly require implementation and initial support and training effort from the service provider, and provide the user with an increased measure of perceived product quality and trust in the organization, due to either receiving personalized advice on how to improve their game or the guarantee that all opponents are competitively legitimate.

4.1.1.5 Idea

An idea can be defined as the most embryonic form of a new product or service. It has a directly relation with the opportunity identification phase and can usually be translated in a high-level view of the envisioned solution for the problem creating the opportunity.

The idea is also an important element of the NCD as it's definition should achieve the highest business value possible, influencing the future success and overall health of the business (Koen *et al.*, 2002).

Therefore, this project's core idea is to adopt a machine learning approach on which the implementation an autonomous Poker-playing agent will be based.

4.1.1.6 Concept definition

Concepts are well-defined ideas which include a written and visual description that includes primary features, customer benefits and some understanding of the solutions' technological requisites. These represent the final element of the NCD and provide the exit to NPD (Koen *et al.*, 2002).

As such, this project's concept consists of the development of an autonomous Poker-playing agent which is coupled with a user-interface enabling users to match against it.

4.1.2 Functional identification and analysis

The first step in the Quality Function Deployment (QFD) technique is the analysis of the customer value chain, fully depicting the customer and translating his requirements into a product design. The advantage obtained from this technique is the representation of the separate customer groups the solution is intended to serve (McGraw-Hill, 2003).

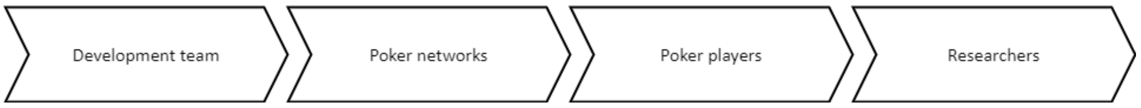


Figure 20 – Customer value chain

In this stage of the VA process a product analysis will be conducted accounting for the main requirements and functions it makes available to the customer.

This process is divided in 2 parts, with the first being the functional identification in which the product's main requirements and functions are identified. A function can be defined as the esteemed value it transmits to the client, and the set of functions identified in this part of the VA process are the parts of the product which will ultimately contribute to its esteemed value (Rich and Holweg, 2000) and therefore to its overall viability.

These functions are described by the following tree diagram in Figure 20:

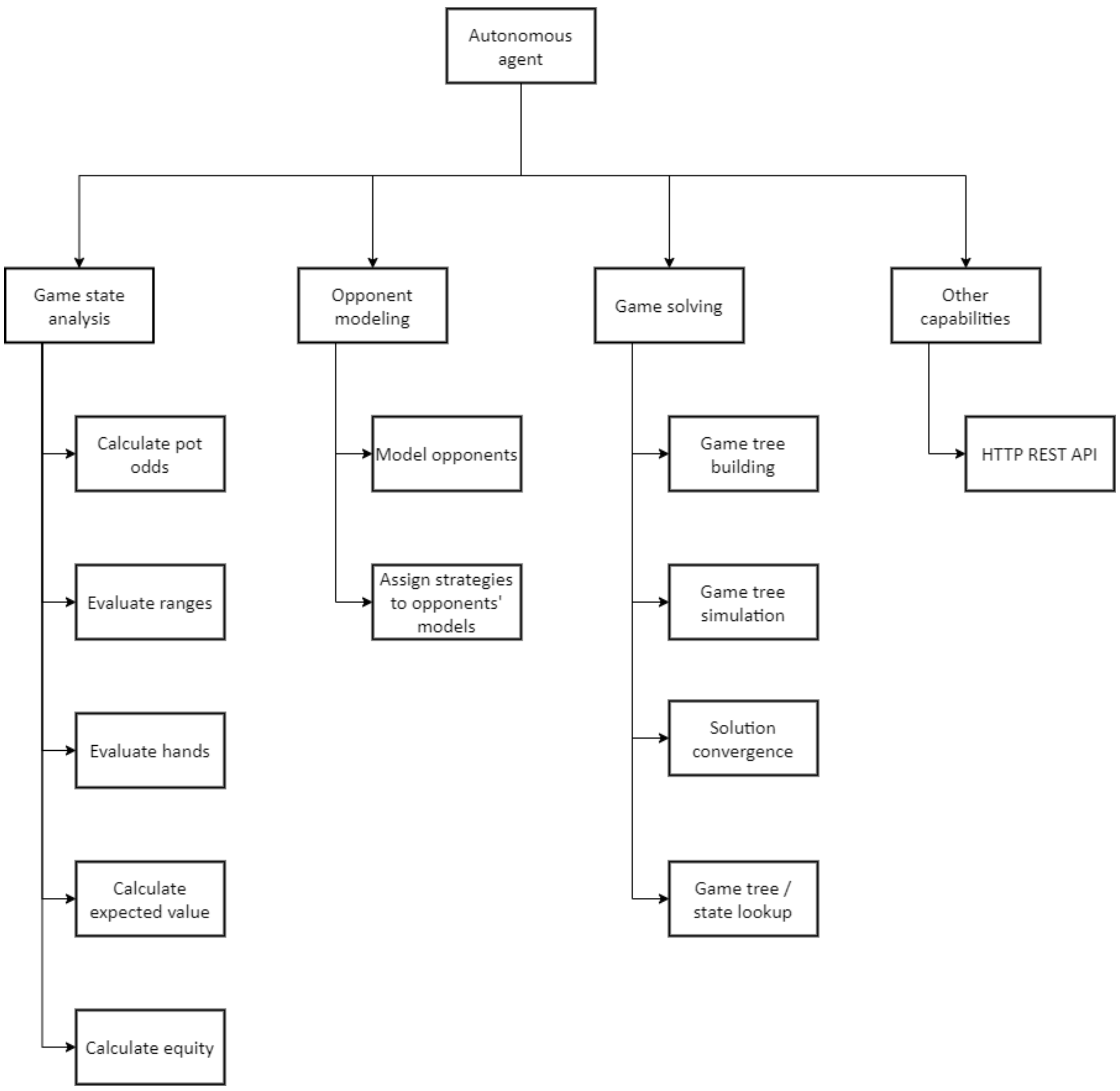
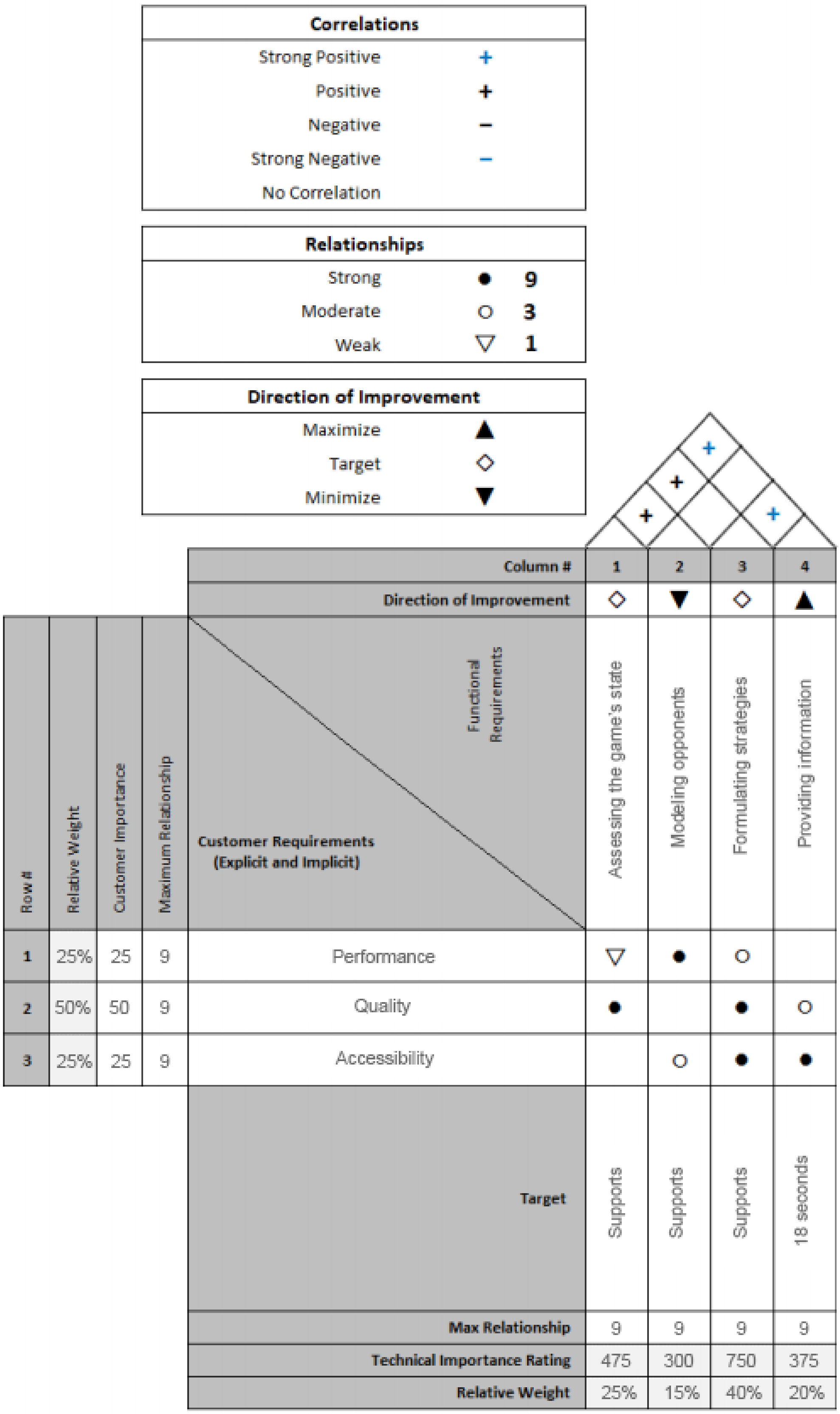


Figure 21 – Functionality tree diagram

The following part of this process consists of the functional analysis, in which functions are comparatively evaluated. The House of Quality technique will be applied in order to translate the customer's desired attributes into engineering requisites, one of the most important results of the QFD. Using the aforementioned technique, quality and functionality requirements are compared and the relationships between them evaluated in order to identify positive and negative relations.

The functional requirements presented in Figure 22 are a subset of the requirements listed in chapter 5.1.1.



Correlations	
Strong Positive	+
Positive	+
Negative	−
Strong Negative	−
No Correlation	

Relationships		
Strong	●	9
Moderate	○	3
Weak	▽	1

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼

	Column #	1	2	3	4
	1		+	+	+
	2				
	3				+

				Column #	1	2	3	4
				Direction of Improvement	◇	▼	◇	▲
Row #	Relative Weight	Customer Importance	Maximum Relationship	Functional Requirements / Customer Requirements (Explicit and Implicit)	Assessing the game's state	Modeling opponents	Formulating strategies	Providing information
1	25%	25	9	Performance	▽	●	○	
2	50%	50	9	Quality	●		●	○
3	25%	25	9	Accessibility		○	●	●
				Target	Supports	Supports	Supports	18 seconds
				Max Relationship	9	9	9	9
				Technical Importance Rating	475	300	750	375
				Relative Weight	25%	15%	40%	20%

Figure 22 – House of Quality

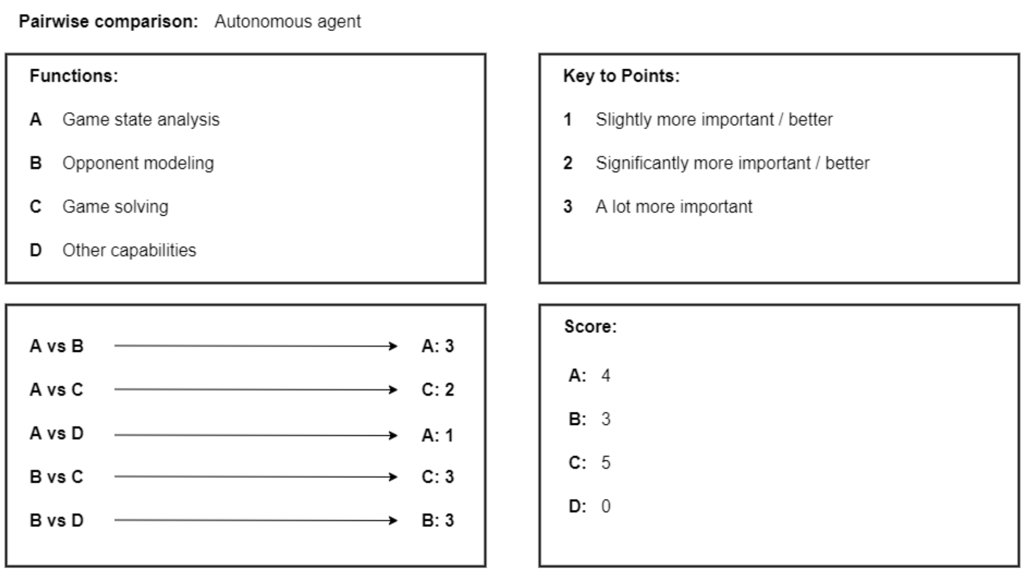
The following table describes the customer and functional requirements considered in order to build the House of Quality.

Table 5 – Requirements considered for the House of Quality analysis

Type	Requirement	Customer Importance
Customer	Performance	25
Customer	Quality	50
Customer	Accessibility	25
Functional	Assessing the game's state	-
Functional	Modeling opponents	-
Functional	Formulating strategies	-
Functional	Providing information	-

From the House of Quality analysis (Figure 22) we can derive that the most important requirement from a user's standpoint is "Formulating strategies" – the products capability to formulate valid, profitable in-game sequences of actions, in accordance to the game's state. The capacity in which the product interprets of the latter is the second most important requirement, with the strategy formulation being dependent on this.

Having assessed all the product's functionalities, a pairwise comparison is conducted in order to rank them by importance. When all pairs are compared, each pair's resulting sum is calculated, with the one which obtains the highest value being considered as the most relevant function for the project, and the main one to be taken into consideration along the development process (Rich and Holweg, 2000).


Pairwise comparison: Autonomous agent

Functions:

A	Game state analysis
B	Opponent modeling
C	Game solving
D	Other capabilities

Key to Points:

1	Slightly more important / better
2	Significantly more important / better
3	A lot more important

A vs B	**A: 3**
A vs C	**C: 2**
A vs D	**A: 1**
B vs C	**C: 3**
B vs D	**B: 3**

Score:

A:	4
B:	3
C:	5
D:	0

Figure 23 – Pairwise comparison

The results demonstrated by the pairwise comparison show that the most important function of the autonomous agent is that of “Game solving”, followed by “Game state analysis”. These results correlate directly with those obtained in the House of Quality analysis, allowing us to objectively conclude that these are the requirements in which the main focus of this project lies.

4.1.3 Alternatives

In this stage of the value analysis process, tools that enable the development of the product being analyzed are presented. One of the provided alternatives must be selected, taking into consideration not only the most meaningful requirements derived from the previous chapter, but also the production cost to value offered ratio.

A series of alternatives for the implementation and fulfillment of this product’s main requirements have been described in detail in chapter 2.4 and 0, and mainly consist of approaches of the following genre:

- Rule based (2.4.1);
- Evolution based (2.4.2);
- Simulation based (2.4.3);
- Equilibrium based (2.4.4);
- Counterfactual regret minimization (2.4.5).

4.1.4 Analysis and Evaluation

The following phase to be conducted in the value analysis procedure is the analysis, evaluation and decision of the tool which concedes the most value to the consumer at the lowest cost to the provider.

For the means of this VA phase, the Analytical Hierarchy Process (AHP) will be utilized. This is a multi-criteria decision support method initially defined by Thomas L. Saaty (1984) with the purpose of comparing different tools defined in the alternative creation phase according to a set of criteria in order to assess which will be the better tool to use in the project.

Being a discreet multi-criteria decision method, the AHP allows a problem to be divided along multiple hierarchies in order to facilitate the evaluation and decision process, ultimately reaching a decision.

The criteria for this study’s product are the following:

- Implementation – the ease with which the theoretical concepts can be implemented into a fully-fledged application;
- Hardware requirements – in terms of RAM and hard-drive space;
- Time elapsed – how long the application takes before demonstrating the calculated results.

The aggregated criteria and alternatives form the hierarchical decision tree contained in the following figure.

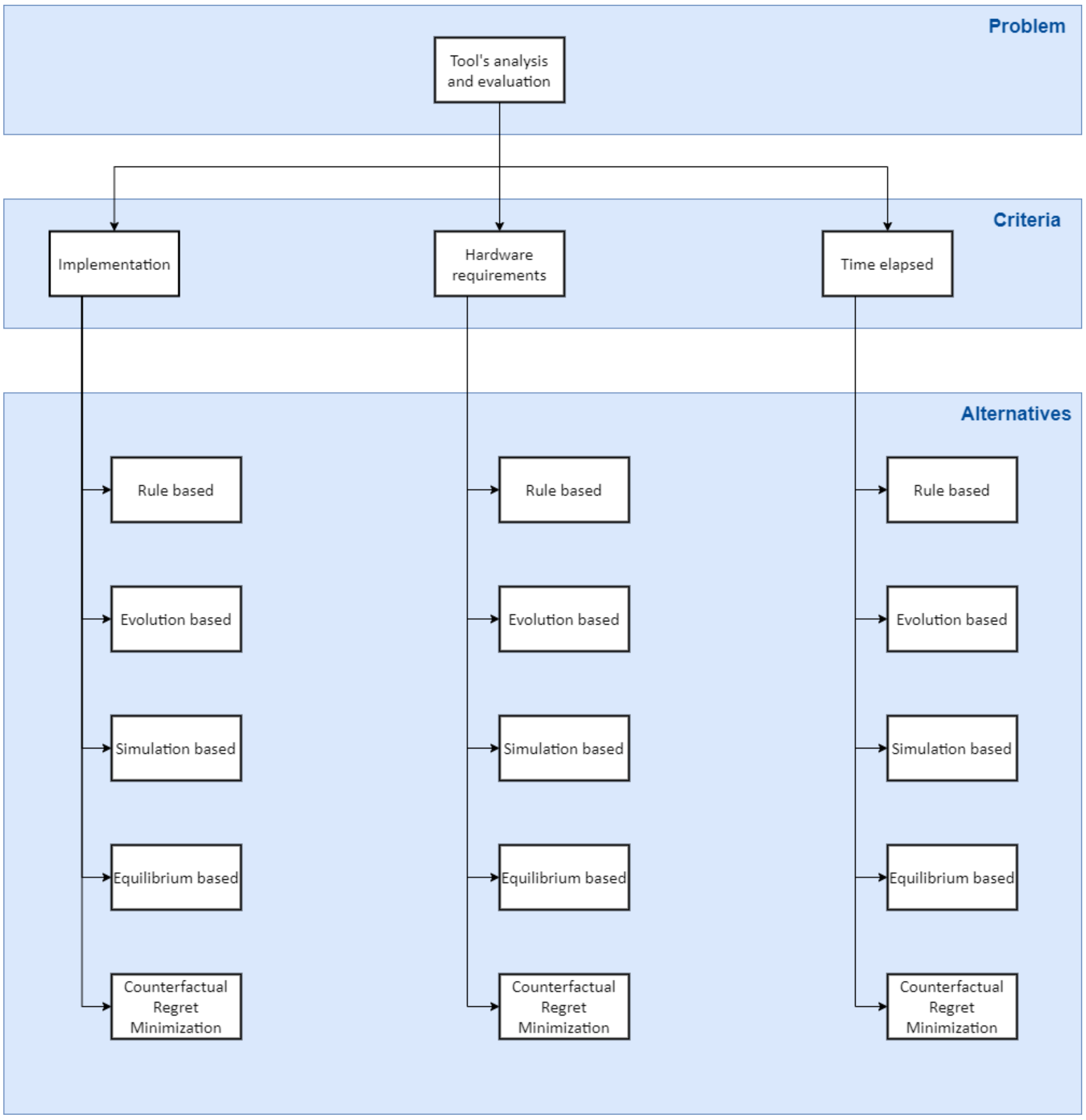


Figure 24 – Hierarchical decision tree

After defining the criteria and building the hierarchical tree for this phase's assessment, the AHP commands the elaboration of a comparison matrix to establish priority between the criteria, according to the method's fundamental scale.

Table 6 – Fundamental scale of the Analytical Hierarchical Process (Saaty, 1984)

Importance on an absolute scale	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
3	Moderate importance of one over another	Experience and judgement lightly favor one activity over another
5	Essential or strong importance	Experience and judgment strongly favor one activity over another
7	Very strong importance	An activity is strongly favored and its dominance demonstrated in practice
9	Extreme importance	The evidence favoring one activity over another is of the highest possible order of affirmation
2, 4, 6, 8	Intermediate values between the two adjacent judgements	When compromise is needed

Considering the fundamental scale's values, priorities are then attributed for each comparison set of the criteria.

Table 7 – AHP's criteria matrix

	Implementation	**Hardware requirements**	**Time elapsed**
Implementation	1	5	9
Hardware requirements	$\frac{1}{5}$	1	3
Time elapsed	$\frac{1}{9}$	$\frac{1}{3}$	1
Sum	$\mathbf{\frac{59}{45}}$	$\mathbf{\frac{19}{3}}$	**13**

Each column's sum is calculated in order to normalize the matrix and posteriorly calculate a mean, allowing for the criteria to be ordered by level of importance.

What follows is the attribution of an approximate level of relative priority to the normalized matrix.

Table 8 – AHP's criteria matrix normalized and with approximate relative priority

	Implementation	**Hardware requirements**	**Time elapsed**	**Priority**
Implementation	$\frac{45}{59}$	$\frac{15}{19}$	$\frac{9}{13}$	0.75
Hardware requirements	$\frac{9}{59}$	$\frac{3}{19}$	$\frac{3}{13}$	0.18
Time elapsed	$\frac{5}{59}$	$\frac{1}{19}$	$\frac{1}{13}$	0.07

The results define the following criteria priority:

1. Implementation with 75% priority;
2. Hardware requirements with 18% priority;
3. Time elapsed with 7% priority;

The resulting eigenvector is the following:

$$\begin{bmatrix} 0.75 \\ 0.18 \\ 0.07 \end{bmatrix}$$

A priority consistency ratio can be calculated in order to evaluate the consistency of the results by comparing a random index with the consistency index. The resulting consistency ratio value must be < 0.1 (10%) for the priorities to be considered consistency (Saaty, 1984; Nicola, 2019).

$$CR = \frac{CI}{RI} \tag{6}$$

, where CR is the consistency ratio, CI is the consistency index and RI is the random index.

Table 9 – Random Index (RI) values defined by the National Laboratory of Oak Ridge (Nicola, 2019)

n	1	2	3	4	5	6	7	8	9	10
RI	0,00	0,00	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49

In order to obtain the consistency ratio (CR) one must first calculate the consistency index (CI), which can be accomplished through the following formula:

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{6}$$

, where CI is the consistency index, λ_{max} is the eigenvalue and n is the number of evaluated criteria.

The eigenvalue is calculated based on the mean values of the previously obtained eigenvector, which was in turn obtained from the initial criteria matrix. These are then divided by the values of the eigenvector that results from the normalized matrix.

$$Criteria\, matrix\ \times Eigenvector \cong\ \lambda_{max}\ \times Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 5 & 9 \\ \frac{1}{5} & 1 & 3 \\ \frac{1}{9} & \frac{1}{3} & 1 \end{bmatrix} \times \begin{bmatrix} 0.75 \\ 0.18 \\ 0.07 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.75 \\ 0.18 \\ 0.07 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 2.28 \\ 0.54 \\ 0.21 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.75 \\ 0.18 \\ 0.07 \end{bmatrix}$$

$$\Leftrightarrow\ \lambda_{max} \cong 3.01$$

Knowing the λ_{max} value the CI and CR can now be calculated, considering $n = 3$ and $RI = 0{,}58$.

$$CI = \frac{\lambda_{max} - n}{n - 1} =\ 0.005 \qquad\qquad CR = \frac{CI}{RI} = 0.008$$

With the obtained value being inferior to 0.1 or 10%, the priority values are considered to be consistent. Having validated this, a series of pairwise comparisons that intend to measure the relative importance of the associated alternatives by building a comparison matrix for each of these and repeating the aforementioned procedure. The consistency of the obtained results is demonstrated on chapter 0, with all of the criteria having proved to be consistent.

4.1.4.1 Implementation

Table 10 – AHP's criteria matrix for the Implementation criteria

Implementation	**Rule based**	**Evolution based**	**Simulation based**	**Equilibrium based**	**Counterfactual Regret Minimization**
Rule based	1	1	1	1	1
Evolution based	1	1	1	$\frac{1}{3}$	1
Simulation based	1	1	1	1	1
Equilibrium based	1	3	1	1	1
Counterfactual Regret Minimization	1	1	1	1	1
Sum	**5**	**7**	**5**	$\mathbf{\frac{13}{3}}$	**5**

Table 11 – AHP’s criteria matrix normalized and with approximate relative priority for the Implementation criteria

Implementation	**Rule based**	**Evolution based**	**Simulation based**	**Equilibrium based**	**Counterfactual Regret Minimization**	**Priority**
Rule based	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{3}{13}$	$\frac{1}{5}$	**0.20**
Evolution based	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{1}{13}$	$\frac{1}{5}$	**0.15**
Simulation based	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{3}{13}$	$\frac{1}{5}$	**0. 20**
Equilibrium based	$\frac{1}{5}$	$\frac{3}{7}$	$\frac{1}{5}$	$\frac{3}{13}$	$\frac{1}{5}$	**0.25**
Counterfactual Regret Minimization	$\frac{1}{5}$	$\frac{1}{7}$	$\frac{1}{5}$	$\frac{3}{13}$	$\frac{1}{5}$	**0. 20**

4.1.4.2 Hardware requirements

Table 12 - AHP's criteria matrix for the Hardware Requirements criteria

Hardware Requirements	**Rule based**	**Evolution based**	**Simulation based**	**Equilibrium based**	**Counterfactual Regret Minimization**
Rule based	1	9	9	9	9
Evolution based	$\frac{1}{9}$	1	3	5	7
Simulation based	$\frac{1}{9}$	$\frac{1}{3}$	1	3	5
Equilibrium based	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{1}{3}$	1	1
Counterfactual Regret Minimization	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	1	1
Sum	$\mathbf{\frac{13}{9}}$	$\mathbf{\frac{1121}{105}}$	$\mathbf{\frac{203}{15}}$	**19**	**23**

Table 13 - AHP's criteria matrix normalized and with approximate relative priority for the Hardware Requirements criteria

Hardware Requirements	Rule based	Evolution based	Simulation based	Equilibrium based	Counterfactual Regret Minimization	Priority
Rule based	$\frac{9}{13}$	$\frac{945}{1121}$	$\frac{135}{203}$	$\frac{9}{19}$	$\frac{9}{23}$	**0.60**
Evolution based	$\frac{1}{13}$	$\frac{105}{1121}$	$\frac{45}{203}$	$\frac{5}{19}$	$\frac{7}{23}$	**0.18**
Simulation based	$\frac{1}{13}$	$\frac{35}{1121}$	$\frac{15}{203}$	$\frac{3}{19}$	$\frac{5}{23}$	**0.10**
Equilibrium based	$\frac{1}{13}$	$\frac{21}{1121}$	$\frac{5}{203}$	$\frac{1}{19}$	$\frac{1}{23}$	**0.08**
Counterfactual Regret Minimization	$\frac{1}{13}$	$\frac{15}{1121}$	$\frac{3}{203}$	$\frac{1}{19}$	$\frac{1}{23}$	**0.04**

4.1.4.3 Time elapsed

Table 14 - AHP's criteria matrix for the Time Elapsed criteria

Time Elapsed	Rule based	Evolution based	Simulation based	Equilibrium based	Counterfactual Regret Minimization
Rule based	1	9	9	9	9
Evolution based	$\frac{1}{9}$	1	3	5	7
Simulation based	$\frac{1}{9}$	$\frac{1}{3}$	1	3	5
Equilibrium based	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{1}{3}$	1	1
Counterfactual Regret Minimization	$\frac{1}{9}$	$\frac{1}{7}$	$\frac{1}{5}$	1	1
Sum	$\mathbf{\frac{13}{9}}$	$\mathbf{\frac{1121}{105}}$	$\mathbf{\frac{203}{15}}$	**19**	**23**

Table 15 - AHP's criteria matrix normalized and with approximate relative priority for the Time Elapsed criteria

Time Elapsed	Rule based	Evolution based	Simulation based	Equilibrium based	Counterfactual Regret Minimization	Priority
Rule based	$\frac{9}{13}$	$\frac{945}{1121}$	$\frac{135}{203}$	$\frac{9}{19}$	$\frac{9}{23}$	**0.60**
Evolution based	$\frac{1}{13}$	$\frac{105}{1121}$	$\frac{45}{203}$	$\frac{5}{19}$	$\frac{7}{23}$	**0.18**
Simulation based	$\frac{1}{13}$	$\frac{35}{1121}$	$\frac{15}{203}$	$\frac{3}{19}$	$\frac{5}{23}$	**0.10**
Equilibrium based	$\frac{1}{13}$	$\frac{21}{1121}$	$\frac{5}{203}$	$\frac{1}{19}$	$\frac{1}{23}$	**0.08**
Counterfactual Regret Minimization	$\frac{1}{13}$	$\frac{15}{1121}$	$\frac{3}{203}$	$\frac{1}{19}$	$\frac{1}{23}$	**0.04**

4.1.4.4 Overview

In order to calculate which is the most efficient tool, a matrix composed of all of the eigenvalues is created and multiplied by the priorities eigenvector.

$$\begin{bmatrix} 0.20 & 0.6 & 0.6 \\ 0.15 & 0.18 & 0.18 \\ 0.20 & 0.10 & 0.10 \\ 0.25 & 0.08 & 0.08 \\ 0.20 & 0.04 & 0.04 \end{bmatrix} \times \begin{bmatrix} 0.75 \\ 0.18 \\ 0.07 \end{bmatrix} = \begin{bmatrix} \mathbf{0.3} \\ 0.1575 \\ 0.175 \\ 0.2075 \\ 0.16 \end{bmatrix}$$

It is hereby concluded that the best performing tool is Rule based with a 30% priority.

4.1.5 Implementation

Through the AHP methodology, the means for implementing the desired solution have been identified based on the demonstrated criteria. The process concludes that a rule based approach is the most efficient alternative for the implementation of the desired product, and therefore it's design and development shall adhere to the specifications of derived via this procedure.

5 Design

This chapter’s aim is to document the decisions made in the stage where the solution to be developed is in it’s specification stage. The requirements, models and architecture of the application are documented in-depth in this section, along with context and justifications supporting the demonstrated artefacts.

5.1 Requirements

Based on the analysis conducted, a list of requirements was elaborated in order to establish the necessities, responsibilities, and particularities of the solution in development.

Said requirements adhere to the FURPS+ model (Fischer and Jost, 1989), commonly used for classifying software’s functional and non-functional requirements. FURPS+ is an acronym for:

- **Functionality:** Capability, reusability and security;
- **Usability:** Human factors, aesthetics and consistency;
- **Reliability:** Availability, Failure extent, predictability and accuracy;
- **Performance:** Speed, efficiency, capacity and scalability;
- **Supportability:** Extensibility, testability and flexibility.

5.1.1 Functional requirements

A functional requirement specifies a function of a system or component, and is often described via the interactions the system should conduct with users – what input to receive and what output to provide – through User Stories which aid the software development team in understanding what must be produced (Ambler, no date).

Due to the fact that this study's solution consists mostly of functional components which the end-user only experiences in a straight-forward game interface that adheres to the industry's standards, requirements of this typology will instead be presenting regarding mainly the feature set, capabilities, generality and security of the system (Grady, 1987).

The functional requirements taken into consideration are the following:

Identifier	Summary
FR-001 – Game state interpretation	The solution must be able to interpret the state of the game at any given action moment.
FR-002 – Pot odds calculation	The solution must be able to calculate pot odds in any given game state.
FR-003 – Expected value calculation	The solution must be able to calculate expected value in any given game state.
FR-004 – Equity calculation	The solution must be able to calculate equity in any given game state.
FR-005 – Range evaluation	The solution must be able to evaluate ranges in any given game state.
FR-006 – Hand evaluation	The solution must be able to evaluate a hand (hole cards) in any given game state.
FR-007 – Opponent modelling	The solution must be able to model an opponent's strategy in any given game state, given the existence of a sufficient sample size.
FR-008 – Strategy assignment	The solution must be able to attribute pre-modeled strategies to opposing players.
FR-009 – Game tree building	The solution must be able to formulate game trees from any given root node.
FR-010 – Game tree simulation	The solution must be able to simulate play throughout the entirety of a previously generated game tree.
FR-011 – Game tree lookup	The solution must be able to validate if a given game state is known (has been previously calculated).
FR-012 – Simulation convergence	The solution must assure the game tree simulations converge in compliance with NFR-004.
FR-013 – Data storage	The solution stores hand history and / or game tree solutions in the database it is connected to.
FR-014 - Statistics	The solution must calculate and demonstrate basic player statistics on the user-interface.
FR-015 - Matchmaking	The solution must allow players to initiate matches against the autonomous agent.

5.1.2 Non-functional requirements

Functional requirements are supported by non-functional requirements which impose constraints on defined aspects of the system's design or implementation, typically regarding specific categories.

5.1.2.1 Usability

Usability non-functional requirements relate to how the system performs at providing a pleasurable experience to the end user. It was established that this system should:

- Provide an intuitive and ergonomic design which is adjusted to the to the activity it is intended to support;
- Implement ISO 9241's design principles in order to create a user-centered design;

5.1.2.2 Reliability

Reliability non-functional requirements consider multiple factors regarding how dependable the system is in terms of how frequently and severely failures occur and how many resources are necessary in order to recover from them. For this sub-category, a single requirement was gathered:

- Provide state-of-the-art end-to-end encryption, with client-server communication being handled through the HTTPS protocol;

5.1.2.3 Performance

Requirements regarding performance evaluate the measure in which the system's speed, efficiency, throughput and response time satisfy the user's needs (Grady, 1987). Regarding this category, two requirements were established:

- Provide a response in under the action time-out limit defined (15 seconds);
- Maintain performance throughout consecutive games with no restarts;

5.1.2.4 Supportability

Several aspects of the system were considered in order to assure it's adaptability, configurability, compatibility and extensibility, all of which fall under the supportability sub-category of non-functional requirements according to the author Grady, 1987.

- Provide an API that adheres to the HTTP-REST specification;
- Provide a low-latency, bidirectional communication channel using the WebSocket protocol;
- Provide responses in a JSON format;

The non-functional requirements considered by the development team at this point in time are the following:

Identifier	Summary
NFR-001 – API	The solution provides an HTTP-REST API.
NFR-002 – Client-server socket	The solution connects to users through a multilateral communication socket.
NFR-003 – Database connection	The solution is connected to a database with read and write privileges.
NFR-004 – Response time	The solution must consistently provide a response within the established time frame (time-out limit).
NFR-005 – Continuous operation	The solution must be capable of playing against multiple players (not concurrently) without being rebooted or suffering adjustments.
NFR-006 – Usability	Interface design principles that adhere to ISO 9241's specification shall be utilized in order to create a user-centered design.
NFR-007 – Interface design	The solution's interface shall be intuitive, ergonomic and adjusted to the activity it is intended to support.
NFR-008 – Encryption	Users must be able to connect to the solution's server through SSL/TLS (HTTPS).
NFR-009 – Client-server dialogue	The solution shall format its output according to the JSON specification.

5.2 Business Model

By the thorough analysis of the problem as well as the proposed hypothesis, a domain model capable of satisfying the logical as well as technological necessities of the conceptualized solution was assembled.

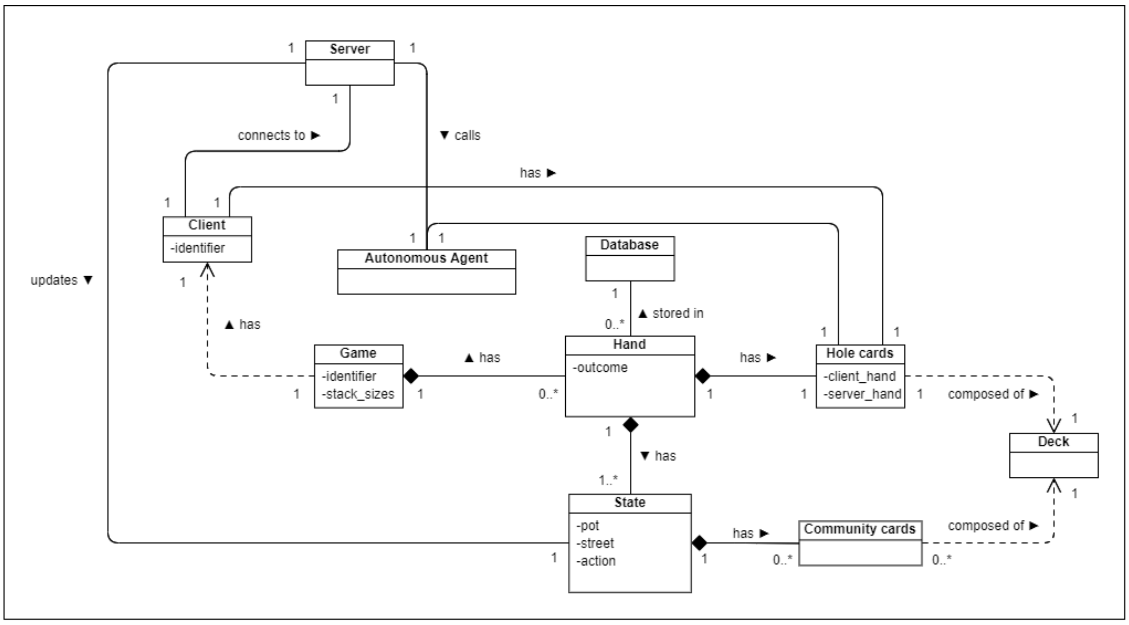


Figure 25 - Domain Model

Said solution is based on a client-server dialog which continuously updates the game's state. Being altered every time a player acts, the state is the information index of each play or hand in Poker terminology. While the state holds every action which occurred at the given moment in each play – including nature's actions, such as the dealing of the community cards – the hand is where each player's hole cards, as well as the outcome of a play, is stored. This decision is supported by the fact that while the game's state varies constantly, the players' hole cards and the outcome of the play is set only once per play and remains unaltered throughout it. Players' hole cards and the board's community cards are a subset of the available cards in the deck.

The information set composed of the server's hand and the current game state is passed onto the autonomous agent in order to retrieve a decision on which to act upon, storing it in the game's state and communicating it to the client in order for the game to proceed.

At the end of each hand, information regarding the autonomous agent's hole cards and the strategies calculated by the autonomous agent are stored in a database for analysis and efficiency purposes.

5.3 Architecture

In order to obtain the performance and integrity standards demanded from a product such as the one in study, it is vital to analyze the infrastructure design possibilities. Standardized practices of software engineering, programming principles and documented patterns were taken into consideration in order to arrive at the design of a solution which meets the project’s requirements to a high standard and is able to be implemented using established, widely utilized and up-to-date technologies. In this section, artefacts describing such a design are presented.

5.3.1 Component diagram

The component diagram displayed in Figure 26 aims to segregate the various levels of the system’s functionality, in this case dividing it in 3 major groups: client, server and database.

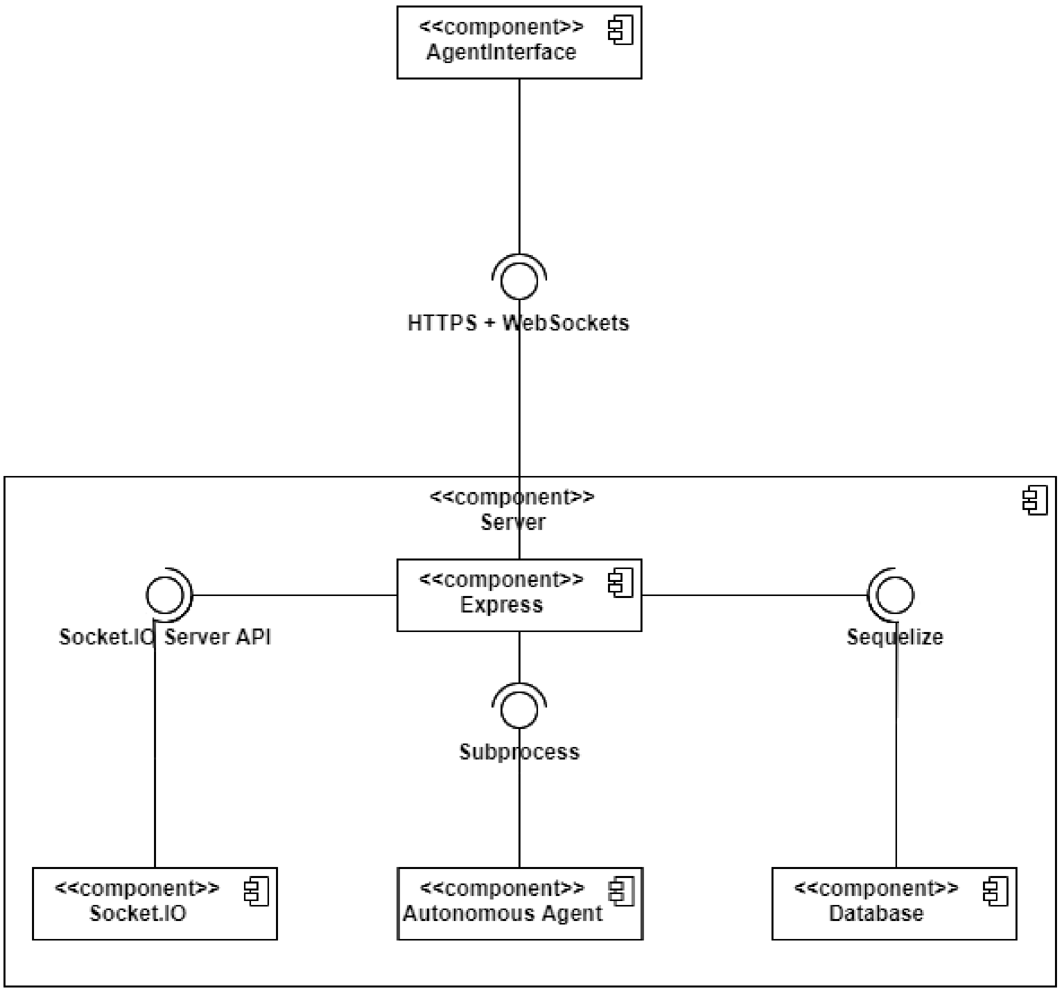


Figure 26 – Solution’s component diagram

Table 16 describes Figure 26’s components.

Table 16 – Solution’s components

Component	Description
Client	The device through which the end user consumes the solution, typically through a web browser or WebView application component.
Server	The device providing the solutions artefacts to the client.
Database	The solution’s database.
Express	The application providing and handling the communication socket established with the client.
Autonomous agent	The application providing the solution with a strategy response to game states which are passed to it.

In a center position of this architecture lies the server, which maintains an open communication channel with the client and mediates the application’s state, sending and receiving it from both the client and the autonomous agent. The server also connects with a database which it uses to query data which may be of use, such as pre-processed strategies for a given state, as well as to store said calculations and results used to measure the agent’s performance.

The autonomous agent’s calculator component resides along the web server, but could be externalized were an API to be made available, as it only needs to receive the game’s state information in order to perform strategy evaluations and formulate a decision.

5.3.2 Deployment diagram

Figure 27 demonstrates the deployment diagram of the components which compose the solution, described singularly in Table 16, assuming the agent’s calculator is not externalized.

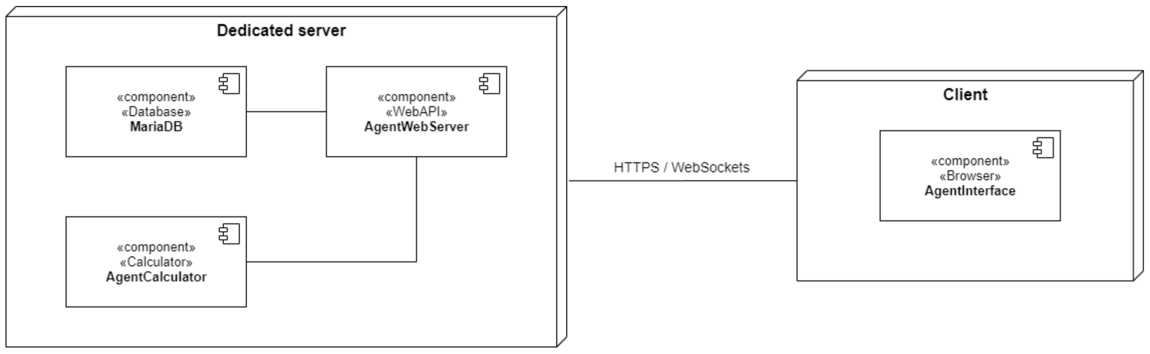


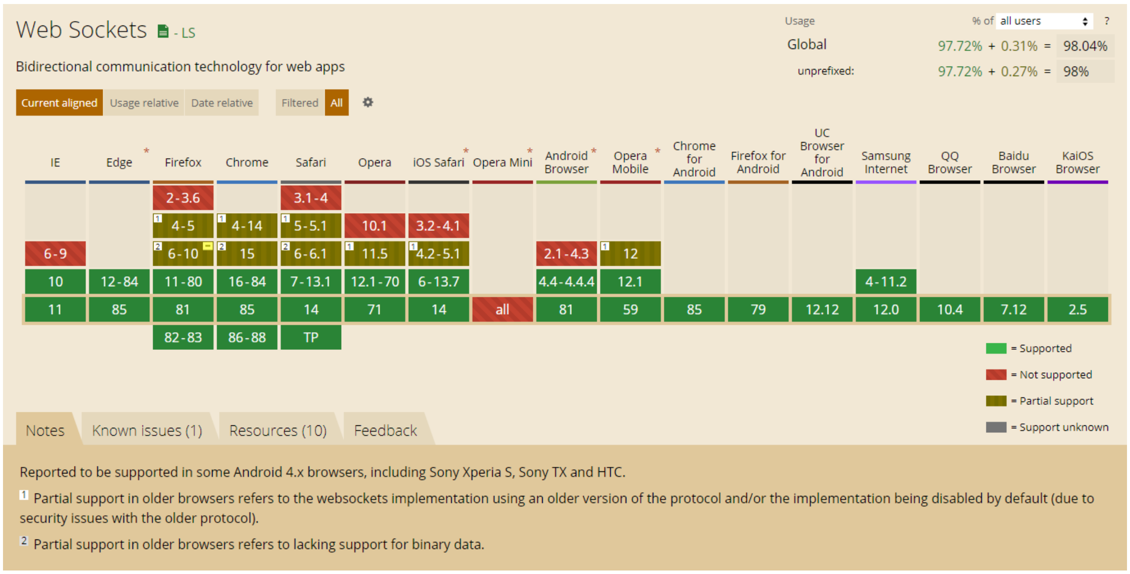
Figure 27 - Deployment diagram

5.3.3 Client-server dialogue

Responding to the application's necessities and in accordance to the component diagram shown in Figure 26, the client-server dialogue is mediated through a socket which establishes and maintains a real-time, bi-directional, event-based and asynchronous communication channel between the client and the server (*Introduction | Socket.IO*, 2020).

While the preferential communications protocol is WebSocket, the library in use falls back to HTTP long-polling in case of lack of support in the client's web-browser, assuring the application's functionality, although with a lower performance standard deriving from the fact that the server is unable to push messages to the client, which will be set to check for updates on a regular interval.

According to data found on the popular web browser compatibility index website caniuse.com, 98% of web browsers support Web Sockets at the time of writing (*Can I use... Web Sockets*, 2020).



Web Sockets - LS

Bidirectional communication technology for web apps

Usage — % of all users — ?

Global: 97.72% + 0.31% = 98.04%

unprefixed: 97.72% + 0.27% = 98%

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge*	Firefox	Chrome	Safari	Opera	iOS Safari*	Opera Mini*	Android Browser*	Opera Mobile*	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
		2-3.6		3.1-4												
		[1] 4-5	[1] 4-14	[1] 5-5.1	10.1	3.2-4.1										
6-9		[2] 6-10	[2] 15	[2] 6-6.1	[1] 11.5	[1] 4.2-5.1		2.1-4.3	[1] 12							
10	12-84	11-80	16-84	7-13.1	12.1-70	6-13.7		4.4-4.4.4	12.1				4-11.2			
11	85	81	85	14	71	14	all	81	59	85	79	12.12	12.0	10.4	7.12	2.5
		82-83	86-88	TP												

🟩 = Supported
🟥 = Not supported
▨ = Partial support
⬛ = Support unknown

Notes | Known issues (1) | Resources (10) | Feedback

Reported to be supported in some Android 4.x browsers, including Sony Xperia S, Sony TX and HTC.

[1] Partial support in older browsers refers to the websockets implementation using an older version of the protocol and/or the implementation being disabled by default (due to security issues with the older protocol).

[2] Partial support in older browsers refers to lacking support for binary data.

Figure 28 - Browser compatibility for Web Sockets as of 2020

The following sequence diagram depicts the message exchange process between a client and the server application when starting and playing a game.

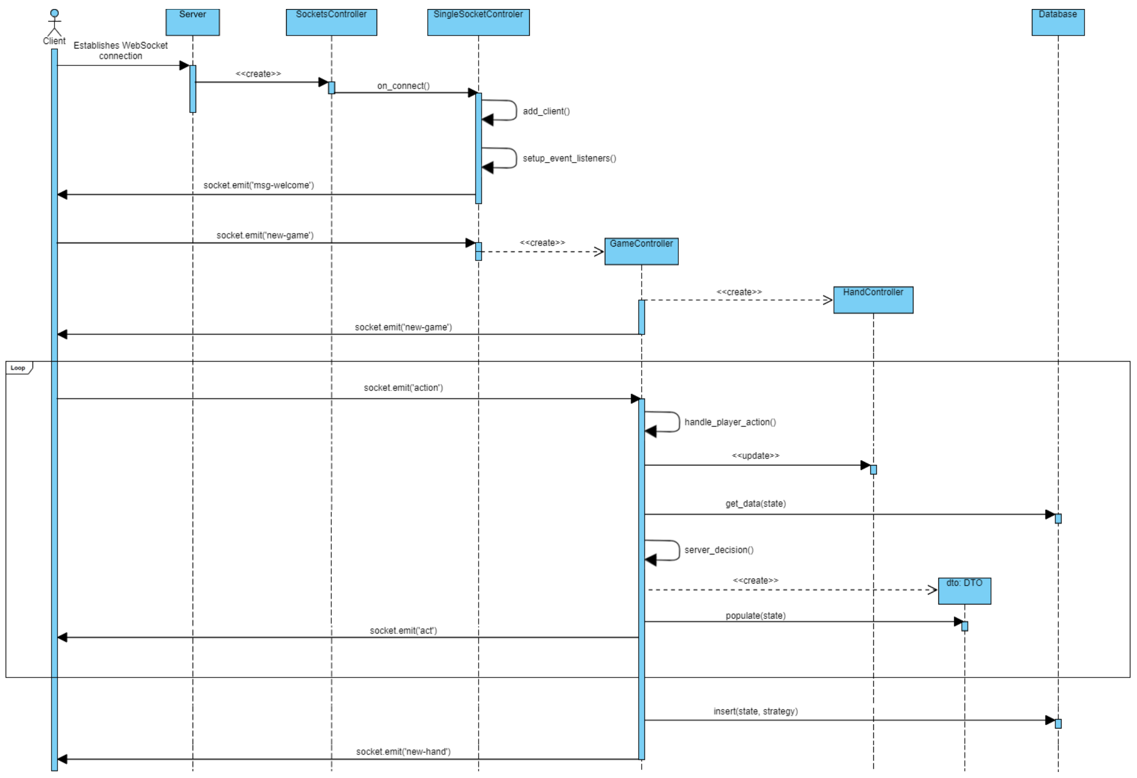


Figure 29 - Client-server dialogue sequence diagram

This sequence diagram describes the processes triggered by events in the following order:

1. A client connects to the server through a web-browser;
2. The client chooses to start a new game;
3. The server application instantiates all necessary objects, sets up the game state and informs the client that the game request was accepted by emitting the "new-game" event to the client;
4. The client act's by emitting an "action" event to the server;
5. The server informs the client it's his turn to act by emitting an "act" event in his direction;

Accompanying every event emitted by the server goes a JSON object containing information regarding every aspect of the current state of the game. The client handles this information, updating it's user interface accordingly in order to provide the end-user with the entirety of the information and options available to players at the given point in the game.

```
{
 "stacks": {
        "client": 9,
        "server": 6
 },
 "table_position": {
        "client": 0,
        "server": 1
 },
 "pot": 5,
 "street": 1,
 "hole_cards": [42, 19],
 "community_cards": {
        "flop": [46, 4, 18],
        "turn": [],
        "river": []
 },
 "action": [
        [{
                        "action": "check-call",
                        "amount": 0
                },
                {
                        "action": "check-call",
                        "amount": 0
        }],
        [{
                "action": "raise",
                "amount": 3
        }],
        [],
        []
 ],
 "outcome": null,
 "num_hands": 0
}
```

Code snippet 3 – JSON object containing game state information (server → client)

The client communicates with the server in a similar manner when emitting the "action" event, which is bundled with a JSON object describing the user's action to the server.

```
{
 "action": 'bet',
 "amount": 3
}
```

Code snippet 4 – JSON object containing user action information (client → server)

The server continuously evaluates the current hand's status, ending the play when certain conditions are met, at which point the "new-hand" event is sent to the client, indicating that the current play is over and the next one has begun.

5.3.4 Design patterns

Software design patterns are formalized best practices which provide established solutions to recurring problems. The use of documentation and design patterns in software development has been proven to have a positive effect on comprehension and therefore maintainability (Wedyan and Abufakher, 2020).

Established sets of patterns such as the Gang of Four (GoF) and Domain Driven Design (DDD) were consulted in an effort to adhere to GRASP’s (General Responsibility Assignment Software Patterns) object-oriented design guidelines.

The following patterns and principles are considered pertinent to the solution’s design:

- **Model-View-Controller:** Segregates an application’s concerns in regards to it’s data structure, management and output. The model is the definition of an object and it’s properties. The view represents a visualization of the model. The controller modifies both the model and the view, being able to alter the model’s properties and update the view.

- **Strategy:** A class’ behavior is altered at runtime based on certain criteria regarding the operation being performed. Different classes are instantiated based on said criteria, implementing a given method of the context object, thus allowing for the same operation to be performed with varying algorithms.

- **Service:** Segregates the application’s business logic from the requirements imposed by the different clients consuming it or different use cases involving complex operations which handle transactional resources.

- **Repository:** Encapsulates the logic required to access the persistence layer, thereby decoupling the database infrastructure from the domain model layer.

- **Data transfer object:** An object that carries data between processes, aggregating multiple resources into a single object which is typically transitioned from the server to the client. It increases performance by reducing the number of calls necessary to fetch data from multiple endpoints in dialogues with remote interfaces where the round-trip between the client and the server is expensive. Similarly, the data can be curated previous to being transitioned, removing sensitive or unnecessary details.

- **Router:** Widely used in web services with CRUD operations available for it’s models, the router decomposes the URI endpoint called into parameters in order to determine which controller and action should handle the request. This pattern is akin to the Facade and Mediator patterns.

6 Development

This chapter describes the development process of the various components of the solution, containing in-depth explanations regarding the decisions made throughout the development process, dissecting the low-level programmatic details of the procedures triggered throughout the flow of the execution, and documenting some of the key particularities of an application such as the one which this study results in.

6.1 Autonomous agent

Having experimented with base implementations of a number of autonomous learning algorithms, such and Counterfactual Regret Minimization (CFR, chapter 2.4.5) and Reinforcement Learning (RL, chapter 2.4.2), it became apparent that the only viable option in order to meet the requirements and particularities of the solution in hand was the implementation of an algorithm based on Fictitious Play (FP), which falls under simulation based approaches for machine learning, discussed in chapter 2.4.3.

This algorithm revolves around simulating every in-game situation available to each player and updating the player's strategy in accordance to the value calculated for all given actions.

In order to formulate a strategy, this component receives the game's state. In particular, the following parameters are necessary:

- Players' stack sizes
- Blinds value (small-blind and big-blind)
- Ante

These parameters are used in order for the solution to calculate the effective stack size and the expected value of each's player's decision at every possible node on this level of the decision tree. This is one of the downsides of this algorithm in comparison to the aforementioned alternatives – it does not simulate or take into consideration the additional decision points in the play, but instead considers the play is to be finalized at the decision point being analyzed.

6.1.1 Strategy formulation

The solution initializes by setting up the data set for the given situation being analyzed: both players' stacks, the effective stack, and an initial guess for each player's range, which we can assume to consist of the top 50% hands for the sake of simplicity.

Posteriorly, the expected value of each action in each of the 1 326 hands each player might have is calculated and stored in a strategy profile.

```
num_iterations = 500
num_hands = 1326
players = ['SB', 'BB']
blinds = { 'SB': 0.5, 'BB': 1 }
stacks = { 'SB': 10, 'BB': 10 }
effective_stack = max(stacks.values())
ranges = { 'SB': Range(0.5), 'BB': Range(0.5) }
ranges_exploitative = { 'SB': Range(0), 'BB': Range(0) }

for i in num_iterations:
 for player in players:
   for hand in range(num_hands):
        opponent = int(not players.index(player))

        # Calculate EV of each action
        freq_call_opponent = ranges[opponent].get_play_frequency()
        equity = equity_calc(hand, range_bb)
        ev = {
               'raise': (1 - freq_call_opponent) * (stacks[player]
                         + blinds['BB'])
                         + (freq_call_opponent * equity * (pot + bet)),
               'check_call':  freq_call_opponent * equity * pot,
               'fold':  stacks[player] - blinds['SB']
        }

        # Find the key with the highest EV
        max_ev = ev.keys()[ev.index(max(ev.values()))]

        # Update exploitative range (Boolean)
        for action in ev.keys():
          ranges_exploitative[player]
          .set_hand_freq(hand, action, int(max_ev == action))

        # Update the overall range
        ranges[player].update(ranges_exploitative[player], i)
```

Code snippet 5 – logic demonstration of the fictitious play algorithm

Repeated iteration of this method converges to a strategy that approximates the Nash equilibrium, and is therefore unexploitable – in other terms, it is not possible for an opponent to have positive expectation versus a player employing this strategy. The opponent's best response strategy is to play the Nash equilibrium as well, with any deviation resulting in a decrease in the expected value of his game.

The outcome of this procedure is a range object for each player, containing every possible hand as key and a frequency figure as value. The algorithm generates the opposing player's game-theory optimal strategy in order to continuously learn how to play against it until an equilibrium is reached.

The method through which the range converges is worthy of mention as it affects not only the solution that the autonomous agent will output, but also the time elapsed to produce it.

```
def updateRange(self, r, n):
  fraction = 1 - (1 / (n + config.alpha))

  for h in range(len(hands)):
    self.card_range = (self.card_range[h] * fraction)
                    + (r.card_range[h] * (1-fraction))
```

Code snippet 6 – convergence algorithm

The parameters of the function are the latest calculated range (r) and the number of the iteration it was calculated in (n), and the idea is to incorporate a significant portion of the initial calculations into the overall strategy, but rapidly decrease the significance with which each iteration affects it. For this purpose, the variable named "fraction" is calculated, being attributed a value which increases in proportion to the number of the current iteration - parameter "n".

Knowing that the percentage of the newly calculated range which is incorporated into the overall strategy is $1 - fraction$, and that $fraction$ is as high as the current iteration, it can be stated that each subsequent iteration of the calculation algorithm has a lesser effect on the final solution than the previous one. The $config.alpha$ parameter is the learning rate hyperparameter, and it assures that the algorithm converges.

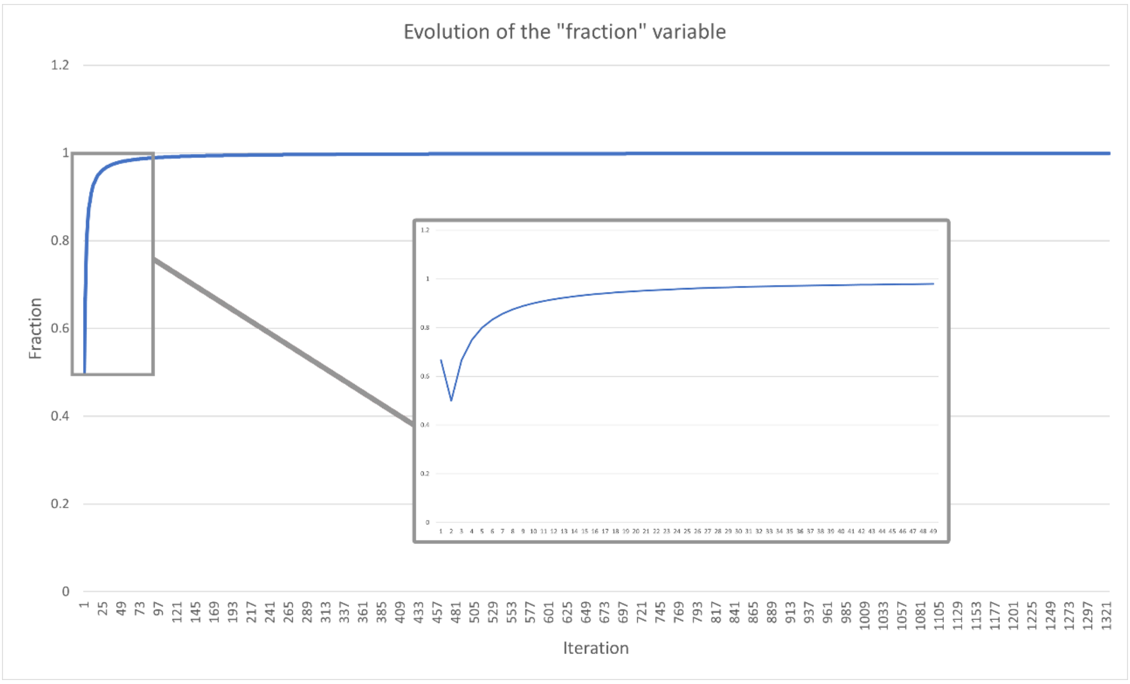
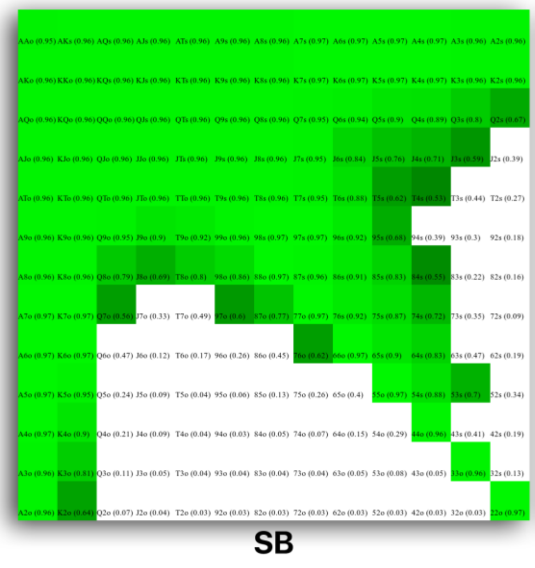


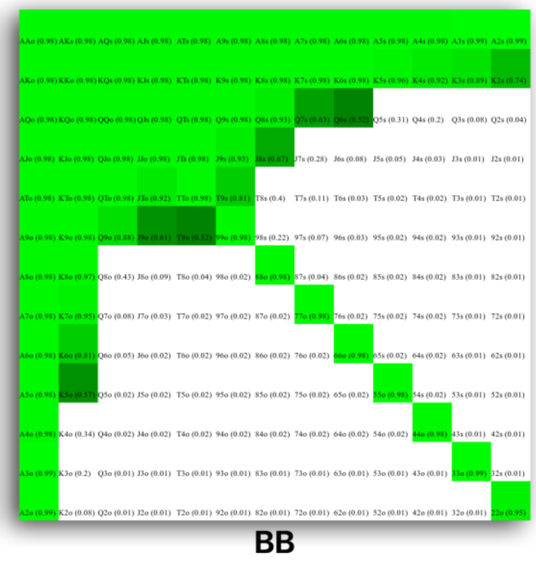
Figure 30 - Evolution of the "fraction" variable in the convergence algorithm

Since it is unpractical to analyze the output of the algorithm in it’s full format, methods were created to aggregate all 1 326 hands into the more common compact format of 169 hands by discriminating hands by suit symmetry, and finally, through scalable vector graphics, generate an image of the compact hand chart, with hands above a certain frequency threshold being colored. This is a significant assistance in terms of visualization and interpretation of the strategy which aids us in the study and further development of the algorithm. Such visualization can be seen in Figure 31.



AAo (0.95)	AKs (0.96)	AQs (0.96)	AJs (0.96)	ATs (0.96)	A9s (0.96)	A8s (0.96)	A7s (0.97)	A6s (0.97)	A5s (0.97)	A4s (0.97)	A3s (0.96)	A2s (0.96)
AKo (0.96)	KKo (0.96)	KQs (0.96)	KJs (0.96)	KTs (0.96)	K9s (0.96)	K8s (0.96)	K7s (0.97)	K6s (0.97)	K5s (0.97)	K4s (0.97)	K3s (0.96)	K2s (0.96)
AQo (0.96)	KQo (0.96)	QQo (0.96)	QJs (0.96)	QTs (0.96)	Q9s (0.96)	Q8s (0.96)	Q7s (0.95)	Q6s (0.94)	Q5s (0.9)	Q4s (0.89)	Q3s (0.8)	Q2s (0.67)
AJo (0.96)	KJo (0.96)	QJo (0.96)	JJo (0.96)	JTs (0.96)	J9s (0.96)	J8s (0.96)	J7s (0.95)	J6s (0.84)	J5s (0.76)	J4s (0.71)	J3s (0.59)	J2s (0.39)
ATo (0.96)	KTo (0.96)	QTo (0.96)	JTo (0.96)	TTo (0.96)	T9s (0.96)	T8s (0.96)	T7s (0.95)	T6s (0.88)	T5s (0.62)	T4s (0.53)	T3s (0.44)	T2s (0.27)
A9o (0.96)	K9o (0.96)	Q9o (0.95)	J9o (0.9)	T9o (0.92)	99o (0.96)	98s (0.97)	97s (0.97)	96s (0.92)	95s (0.68)	94s (0.39)	93s (0.3)	92s (0.18)
A8o (0.96)	K8o (0.96)	Q8o (0.79)	J8o (0.69)	T8o (0.8)	98o (0.86)	88o (0.97)	87s (0.96)	86s (0.91)	85s (0.83)	84s (0.55)	83s (0.22)	82s (0.16)
A7o (0.97)	K7o (0.97)	Q7o (0.56)	J7o (0.33)	T7o (0.49)	97o (0.6)	87o (0.77)	77o (0.97)	76s (0.92)	75s (0.87)	74s (0.72)	73s (0.35)	72s (0.09)
A6o (0.97)	K6o (0.97)	Q6o (0.47)	J6o (0.12)	T6o (0.17)	96o (0.26)	86o (0.45)	76o (0.62)	66o (0.97)	65s (0.9)	64s (0.83)	63s (0.47)	62s (0.19)
A5o (0.97)	K5o (0.95)	Q5o (0.24)	J5o (0.09)	T5o (0.04)	95o (0.06)	85o (0.13)	75o (0.26)	65o (0.4)	55o (0.97)	54s (0.88)	53s (0.7)	52s (0.34)
A4o (0.97)	K4o (0.9)	Q4o (0.21)	J4o (0.09)	T4o (0.04)	94o (0.03)	84o (0.05)	74o (0.07)	64o (0.15)	54o (0.29)	44o (0.96)	43s (0.41)	42s (0.19)
A3o (0.96)	K3o (0.81)	Q3o (0.11)	J3o (0.05)	T3o (0.04)	93o (0.04)	83o (0.04)	73o (0.04)	63o (0.05)	53o (0.08)	43o (0.05)	33o (0.96)	32s (0.13)
A2o (0.96)	K2o (0.64)	Q2o (0.07)	J2o (0.04)	T2o (0.03)	92o (0.03)	82o (0.03)	72o (0.03)	62o (0.03)	52o (0.03)	42o (0.03)	32o (0.03)	22o (0.97)

SB



AAo (0.98)	AKs (0.98)	AQs (0.98)	AJs (0.98)	ATs (0.98)	A9s (0.98)	A8s (0.98)	A7s (0.98)	A6s (0.98)	A5s (0.98)	A4s (0.98)	A3s (0.99)	A2s (0.99)
AKo (0.98)	KKo (0.98)	KQs (0.98)	KJs (0.98)	KTs (0.98)	K9s (0.98)	K8s (0.98)	K7s (0.98)	K6s (0.98)	K5s (0.96)	K4s (0.92)	K3s (0.89)	K2s (0.74)
AQo (0.98)	KQo (0.98)	QQo (0.98)	QJs (0.98)	QTs (0.98)	Q9s (0.98)	Q8s (0.93)	Q7s (0.63)	Q6s (0.52)	Q5s (0.31)	Q4s (0.2)	Q3s (0.08)	Q2s (0.04)
AJo (0.98)	KJo (0.98)	QJo (0.98)	JJo (0.98)	JTs (0.98)	J9s (0.93)	J8s (0.67)	J7s (0.28)	J6s (0.08)	J5s (0.05)	J4s (0.03)	J3s (0.01)	J2s (0.01)
ATo (0.98)	KTo (0.98)	QTo (0.98)	JTo (0.92)	TTo (0.98)	T9s (0.81)	T8s (0.4)	T7s (0.11)	T6s (0.03)	T5s (0.02)	T4s (0.02)	T3s (0.01)	T2s (0.01)
A9o (0.98)	K9o (0.98)	Q9o (0.88)	J9o (0.61)	T9o (0.52)	99o (0.98)	98s (0.22)	97s (0.07)	96s (0.03)	95s (0.02)	94s (0.02)	93s (0.01)	92s (0.01)
A8o (0.98)	K8o (0.97)	Q8o (0.43)	J8o (0.09)	T8o (0.04)	98o (0.02)	88o (0.98)	87s (0.04)	86s (0.02)	85s (0.02)	84s (0.02)	83s (0.01)	82s (0.01)
A7o (0.98)	K7o (0.95)	Q7o (0.08)	J7o (0.03)	T7o (0.02)	97o (0.02)	87o (0.02)	77o (0.98)	76s (0.02)	75s (0.02)	74s (0.02)	73s (0.01)	72s (0.01)
A6o (0.98)	K6o (0.81)	Q6o (0.05)	J6o (0.02)	T6o (0.02)	96o (0.02)	86o (0.02)	76o (0.02)	66o (0.98)	65s (0.02)	64s (0.02)	63s (0.01)	62s (0.01)
A5o (0.98)	K5o (0.57)	Q5o (0.02)	J5o (0.02)	T5o (0.02)	95o (0.02)	85o (0.02)	75o (0.02)	65o (0.02)	55o (0.98)	54s (0.02)	53s (0.01)	52s (0.01)
A4o (0.98)	K4o (0.34)	Q4o (0.02)	J4o (0.02)	T4o (0.02)	94o (0.02)	84o (0.02)	74o (0.02)	64o (0.02)	54o (0.02)	44o (0.98)	43s (0.01)	42s (0.01)
A3o (0.99)	K3o (0.2)	Q3o (0.01)	J3o (0.01)	T3o (0.01)	93o (0.01)	83o (0.01)	73o (0.01)	63o (0.01)	53o (0.01)	43o (0.01)	33o (0.99)	32s (0.01)
A2o (0.99)	K2o (0.08)	Q2o (0.01)	J2o (0.01)	T2o (0.01)	92o (0.01)	82o (0.01)	72o (0.01)	62o (0.01)	52o (0.01)	42o (0.01)	32o (0.01)	22o (0.95)

BB

Figure 31 - Strategy visualization

6.1.2 Hand evaluation

One of the solution's most important components is it's hand evaluation procedure, which allows the agent to calculate the expected value of the multiple strategic choices presented by a given situation. For such purposes, the solution has to be able to calculate a player's equity based on the available information in a certain game state.

With the mathematical and theoretical bases of this process discussed in length in chapter 2.2.5, the focus can now be placed upon the software-engineering, practical standpoint of this endeavor. While it may at first appear that implementing such an algorithm is a feat that can be performed through the simplistic process of enumerating every possibility in a game's state, calculating the winning and drawing odds of each hand and finally the player's equity, the problem in hand is actually non-trivial due to number of reasons which foment debate from a software-engineering point-of-view. This debate relates to the trade-off between size and time in solving a problem, in which it is considered that the amount of time necessary to solve a problem is inversely proportional to the size available to be occupied for solving it. Tree traversals are one example of this, with the amount of space necessary to traverse a tree being $O(\log(n))$[23] if each node is visited a single time, or $O(1)$ if each node is visited 3 times.

As stated by Andrew Pork, the author of PokerStove[24], one of the most popular equity calculators, programmers are faced with a decision when developing an algorithm intended to solve complex problems: whether to create a solution which addresses the problem with a high degree of specificity, or to create a solution which addresses the problem in a more general manner and which may be applied to domains which are similar to but separate from that of the problem in hand. The former option describes a zealous approach, in which the outcome of the development will consist of a complex library which considers specific situations presented by the problem, while the later describes an agnostic approach, which will result in the development of a less complex library with a higher degree of abstraction based on which more complex tools may be developed.

Most if not all the available poker calculation libraries to date used an agnostic approach, which was not suited the performance requirements end users demand of these types of applications. While underlining many of the underlying issues in the zealous approaches, such as size of code base, deteriorated maintenance and susceptibility to failure, an understanding was reached that such an approach was necessary in order to achieve the highest degree of performance, enabling for optimizations across conceptual levels that would be unattainable in an architecturally agnostic approach.

[23] n represents the number of nodes in the tree.
[24] PokerStove is a Poker hand evaluation library written in C++.

Pork describes a practical example in the form of the equity calculation of the hand AA versus 9 opponents whose hole cards are unknown. Through an agnostic approach, every possible scenario (hand combination) would have to be evaluated, of which there are 10^{28} combinations. But by the means of a zealous approach, this number can be reduced to only 133 784 560 – the number of different 7-card hands, which we know are all that is needed to be evaluated in order to equate the hand's equity versus the 9 opponents.

In this same reference, the concept of "subjective all-in equities for full hand distributions" (Pork, 2004) is also explored. In present times and throughout this work, this is more often referred to as a hand versus range evaluation, and it is invaluable to the development of our solution, in which this kind of evaluation is used to determine expected value.

Several attempts were made at developing a proprietary hand versus range evaluator function by enumerating every possible hand the opponent might hold and averaging the equities of every hand versus hand scenario. This immediately proved to be unfeasible, as a single calculation would take several minutes and produce subpar results, and 1 326 of these calculations are necessary per every iteration of the fictitious play algorithm. [25]

With the knowledge that this was a non-trivial, resource intensive process, several hand evaluation libraries were studied in order to select one suitable for our calculation, performance and integration necessities. Table 17 - Hand evaluator libraries summarizes the characteristics and capabilities of the options studied.

Table 17 - Hand evaluator libraries

Library	vs hand	vs range	64 bit	Python 3	Performant
pypoker-eval	✓	✓	✗	✗	?
freepokerdb	✓	✓	✗	✗	?
Deuces / Treys	✗	✗	✓	✓	✓
holdem_calc	✓	✗	✓	✓	✗
pbots_calc	✓	✓	✗	✗	✓

A decision was made to utilize the pbots_calc library, which allowed hand-versus-range and range-versus-range calculations and displayed significantly superior performance in comparison to other libraries testes. This library was originally developed for use in the MIT Pokerbots Competition.

This library was originally developed to be compiled for 32 bit systems. Due to the memory access requirements of our solution, a custom wrapper was developed in order to make this library available for use in a x64 Python runtime.

[25] The fictitious play algorithm studies every possible hand each player might hold. There are 1 326 available starting hands. Assuming there are 2 players, the total number of iterations is $1326 * 2 = 2\,652$.

6.1.3 Performance concerns

With hindering slow initial implementations of the algorithm, it quickly became apparent that resource optimization was an indispensable aspect of the development of this solution.

Starting from a non-software engineering point of view, certain configuration parameters were found to have the potential to significantly impact the performance of computation, the first of which relates to the number of iterations necessary in order for the strategy output to approximate the theoretically correct solution. Questions such as “what is the minimum amount of time necessary to a 100% accurate solution” versus “what is the minimum amount of time necessary to produce a solution with an error margin that doesn’t exceed 2,5%” become pertinent and influence the development process. Empirically, it was found that for simple scenarios such as a 10 big-bling shove-fold game, 100 iterations were enough to produce a viable solution with a reduced error margin, but 200 are recommended in order to consistently produce the theoretically correct solution.

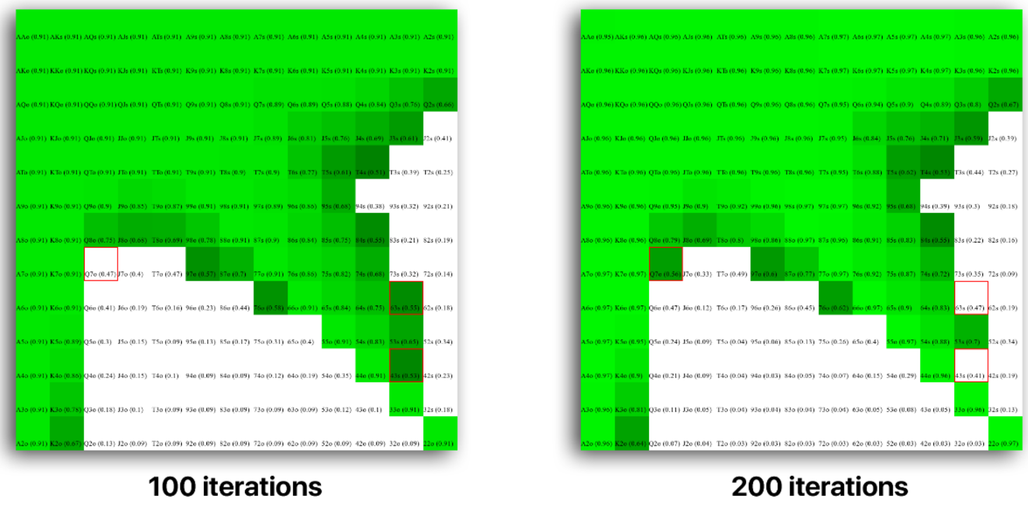


Figure 32 - Differences in solution by number of iterations

One configuration parameter which majorly affects the aforementioned issue is the initial guess provided for each player’s range. The closer the initial guess is to the theoretically correct result, the less the algorithm needs to converge in order to convey the exact solution, which greatly reduces the number of iterations necessary for such an output. While it may appear unfeasible to gear this in our favor, as the correct frequency varies based on the game state input provided and even obtaining an approximation is uneconomical, through the analysis of an aggregate of solutions within particular parameters one could estimate the values between which the solution typically resides, and improve the performance of the algorithm by starting with a more accurate guess.

Performance optimization efforts directed at reducing the time elapsed by the equity calculation procedure in the fictitious play process were introduced – with this being one of the most time-intensive portions of the process. Along with identifying a performant hand evaluator library, posterior improvements were made by storing calculated results in a lookup table. When calculating a certain scenario, the agent would verify if a previous calculation had occurred, using it instead of re-calculating. The cost of this lookup operation is significantly lower than that of the equity calculation, and this method was found to significantly improve performance, due to inefficiencies in the range convergence algorithm, which caused players' ranges to sometimes be unaltered for several consecutive iterations – in which scenario, several iterations would complete without any calculation being performed.

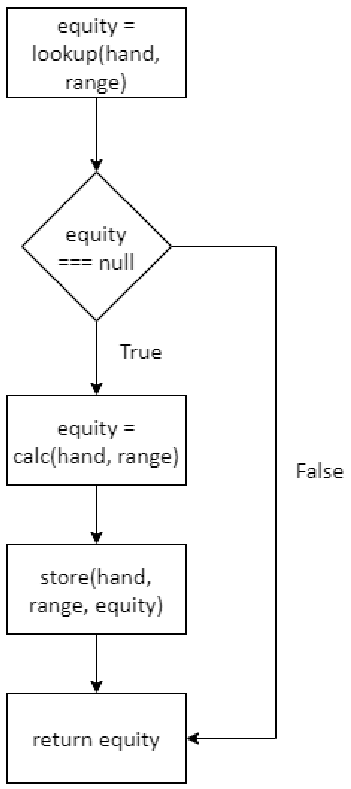

equity = lookup(hand, range)
equity === null
True
equity = calc(hand, range)
False
store(hand, range, equity)
return equity

Figure 33 – Algorithm of a performance optimization through lookup table

This inefficiency is due to the fact that a player is only considered to play actions that have a frequency above a determined threshold, and therefore would not be included in the range even if it's frequency was positively updated in the previous iterations.

It is important to note that in order to obtain the full performance benefits of this methodology, a file with all of the pre-calculated values must be populated and loaded prior to the start of a scenario's computation. This is due to the fact that the operation of loading and writing to this file is resource and time intensive. The resulting file contains several GB of data, which requires a 64-bit Python runtime in order to make this optimization available.

Exhausting the optimization options available for application on the logical layer of the solution, multi-processing was introduced in order to direct all of the computational capacity available to the calculation of the solution. Python provides the process-based threading interface multiprocessing [26] which was utilized for this purpose, allowing concurrent processes to be spawned in an effective and simple to handle manner. By using subprocesses instead of threads, this library allows the bypass of the Global Interpreter Lock[27], unlocking additional performance potential for Pythonic applications.

Due to the way concurrent processes execute and return information, ordered in an often unpredictable manner, the agent's fictitious play algorithm suffered significant adjustments. Processes do not share data between themselves, but only with the parent process, instead being instantiated with the necessary data as parameters, and returning it's output to the parent handler process. This implicates that the convergence algorithm has to reside on the parent process, while the calculation and simulation algorithm have to reside on the child processes. This fact combined with the concurrency of the execution and the unpredictable order in which processes returned data means that processes that take longer to execute were instantiated with a now outdated strategy, and will return a similarly outdated strategy output, which regresses the overall strategy into which it will now be weighted.

The performance loss derived from the previous exposition is acceptable, as the overall performance of the application improves drastically with the use of multi-processing, as shown in Table 18.

Table 18 - Computation time of optimization techniques

Optimization	Description	Execution time 100 iterations
Hand-vs-range algorithm	Proprietary hand-vs-range equity calculation algorithm which found the result by averaging the sum of all hand-vs-hand lookups.	28 minutes
Hand-vs-range algorithm and lookup table	Proprietary hand-vs-range equity calculation algorithm combined with a pre-calculated lookup table of it's own results.	25 minutes
Equity calculator dependency	Equity calculations performed in real time.	15 minutes

[26] https://docs.python.org/2/library/multiprocessing.html

[27] The mechanism used by the CPython interpreter to assure that only one thread executes Python bytecode at a time. (*Glossary — Python 2.7.18 documentation*, 2020)

Equity calculator dependency and lookup table	Hand-vs-range equity lookup table. Algorithm looks for a pre-calculated value, calculates in real-time if unable to find one.	9 minutes
Multiprocessing	Equity calculations performed in real time. Multiprocessing introduced for work to be distributed throughout all cores.	5 minutes
Multiprocessing and lookup table	Hand-vs-range equity lookup table. Algorithm looks for a pre-calculated value, calculates in real-time if unable to find one. Multiprocessing.	3 minutes

In the first optimization techniques attempted, a proprietary preflop hand-versus-range equity calculation algorithm was utilized. As described in chapter 6.1.2, associated with this methodology is a performance figure that is lower than that achieved by using a state of the art equity calculator. The techniques that follow involved pre-calculating equities in order to reduce the time of calculation, which was the most significant portion of every iteration. The later techniques combine the pre-calculated equities with multiprocessing, representing the best performance figure obtained.

6.2 Client

Due to the low performance requirements of a cards-game playing application, the decision was made to implement the client as a Web Application, through the use of HTML5 and JavaScript technologies. Libraries such as jQuery and Socket.IO were used in order to complement the native capabilities of such technologies and accelerate the development process.

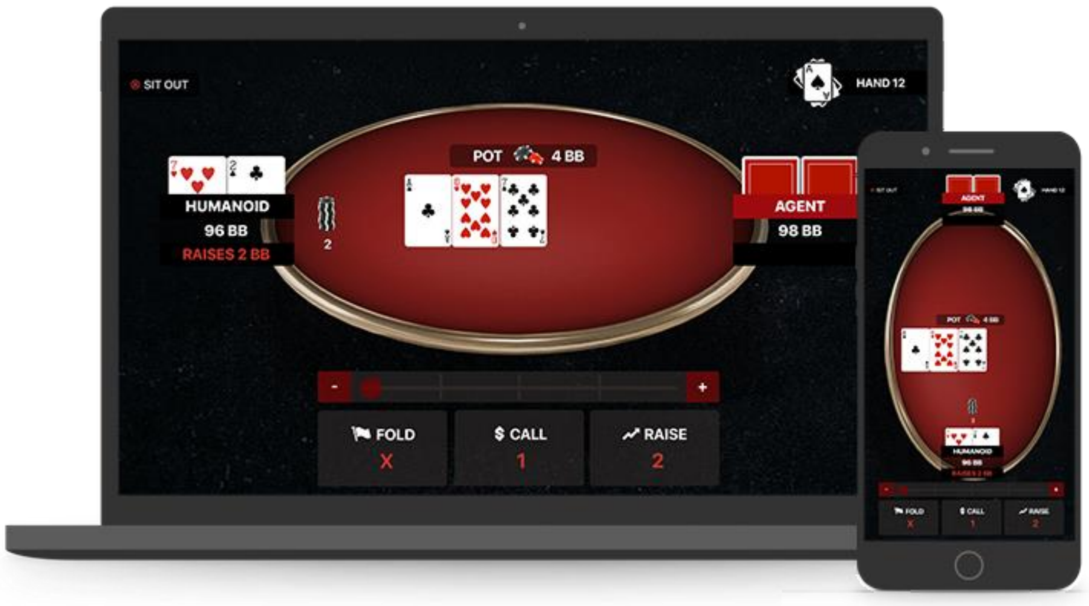


Figure 34 - Game client on desktop and mobile devices

A web application manifest was used in order to approximate the application's behavior with that a user expects from a native application, allowing for possibilities such as adding the application to the home screen, capable of being executed without the user realizing the resort to a web-browser, as it runs in a web-view only instance.

Were the application to be introduced to Google's PlayStore or Apple's App Store, simple applications with a *<iframe>* component occupying the whole screen's width and height could easily be developed, compiling this application into a native format. Methodologies such as this is what frameworks such as Cordova resort to.

In packaging the application for distribution, attention was paid to ensuring cross-device compatibility as well as performance. As such, Gulp was used as a toolkit, aiming to standardize the build process of the application.

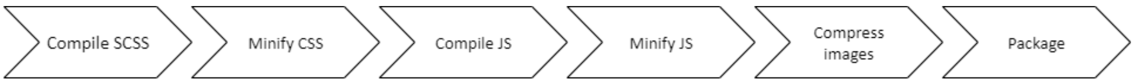


Figure 35 - gulp.js distribution process

Table 19 describes each of the steps shown in Figure 35 in-depth.

Table 19 - gulp.js distribution process

Process	Description
Compile SCSS	Uses `gulp-sass` to compile the feature-rich version of CSS the application was developed in into standardized, cross-browser compatible CSS.
Minify CSS	Uses `gulp-clean-css` to concatenate all CSS files into one and compresses. Saves disk space and bandwidth, improving page-load times.
Compile JS	Uses `gulp-babel` to compile ECMAScript 2015+ into backwards compatible JavaScript. This assures maximum browser and device compatibility.
Minify JS	Uses `gulp-babel` to compress the compiled JS files, saving disk space and bandwidth and improving page-load times.
Compress images	Uses `gulp-imagemin` to compress all images in order to save disk space and bandwidth. Improves page-load times.
Package	Moves the compiled artefacts into a directory segregated from that where the solution artefacts are contained. This is ultimately contains the version of the application intended for distribution.

The compilation procedure the solution undergoes eases development operations, making it more susceptible for testing and deployment in an automated and controlled manner, using the likes of GitHub CI[28] or GitLab CI[29], or setting up a local Jenkins[30] instance.

6.3 Server

The server is the central point of the solution's operation, allowing the user to play against the autonomous Poker agent, invoking it in order to formulate strategies to respond to the player's actions, and managing the database in order to log results and to fetch or store strategies. It is the mediator of the game's state, effectively managing all variables which compose it and serving them to both the client and the autonomous agent.

[28] https://docs.github.com/en/free-pro-team@latest/actions/guides/about-continuous-integration
[29] https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/
[30] https://www.jenkins.io/

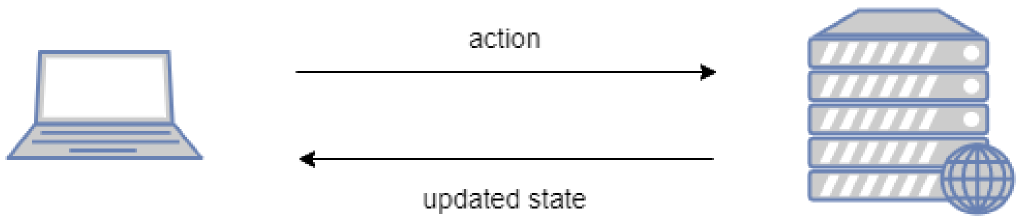


Figure 36 - Client-server communication diagram

Both of these components are limited to interpreting a given game state, having no standalone ability to manage it:

- The client communicates an action to the server, which the server validates and replies with the refreshed game state, given the action received;
- The server communicates with the autonomous agent, to which the agent replies with a strategy upon which the server acts, updating the game state and emitting it to the client.

In order to obtain a gaming experience such as the one players have grown used to in the standardized software of most Poker networks, a socket connection is provided, serving a low-latency, bi-directional communication channel between the client and the server.

The technology selected for the development of this component was NodeJS, with an Express acting as the web-server which hosts the gaming application client and Socket.IO as a socket management library. Both the client and the server API provided by the Socket.IO library are used. Sequelize was used to create an object-relational map between the server's data model and the database. This dependency provides a simplified and standardized method of creating and managing the server's interaction with the database.

The end result is a robust game state management component which is capable of managing a number of separate games with distinct clients simultaneously, capped however by the amount of processing necessary to formulate strategic responses to a single game, or otherwise the amount of storage space necessary in order to store pre-calculated solutions.

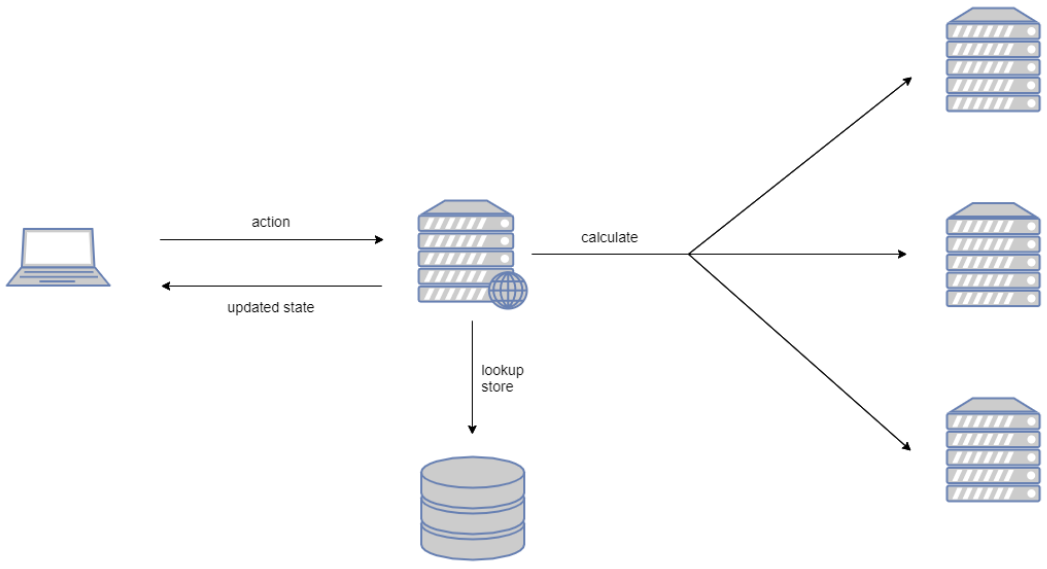


Figure 37 - Network diagram of an communication scenario with an improved setup

One possibly viable solution to the aforementioned complication is the separation of the server component throughout multiple machines, as demonstrated in Figure 37. In such a scenario, a single server, or multiple servers behind a load balancer, would serve the client's frontend application, externalizing the autonomous agent and database components. In the event of a calculation having to be performed, one of the autonomous agent servers – possibly the one with the least load at the time – would be instantiated in order to generate a response. This setup is potentially scalable, and would allow the solution to serve multiple clients at once without the resort to pre-computed solutions.

6.4 Dependencies

The solution depends on a number of external packages, which are documented on Table 20. While package managers were used – namely NPM[31] for the client and server components, and PIP[32] for the autonomous agent – a list of all used packages is due in order to credit the authors which greatly assisted the development of this study.

[31] NPM – Node package manager.
[32] PIP – Python's de facto standard package-management system. According to the author, PIP is an acronym which stands for "Pip installs packages".

Table 20 - Dependencies used for the development of this solution

Artefact	Description
Node.JS	JavaScript run-time environment.
Express	Web server framework.
Socket.IO	JavaScript library enabling bidirectional communication between web clients and servers through the use of WebSockets.
Sequelize	Object-relational mapper for integration on Node.JS applications and support for Postgres, MySQL, MariaDB, SQLite and Microsoft SQL Server databases.
jQuery	JavaScript library facilitating DOM tree traversal and manipulation.
Gulp.js	JavaScript toolkit used to enhance the development workflow.
numpy	Python library supporting mathematical and data-structure manipulation.
pickle	Python module for serializing and de-serializing Python’s object structure.
pbots_calc	Equity calculator for the game of Poker.

7 Experimentation and evaluation

In order to assure that the developed solution meets pre-defined quality standards, a set of tests will be utilized in order to measure the degree to which it fulfills the requirements in terms of functionality and usability.

This chapter describes the evaluation methodology and the indicators used to verify the hypothesis proposed in chapter 1.3, discusses the results and proposes possible future improvements.

7.1 Experimentation methodology

The established evaluation methodology for testing the hypothesis described in chapter 1.3 is based on the measurement of how the solution performs on a series of aspects which will ultimately dictate the degree to which the functional and non-functional requirements have been met.

Considering the ideal solution as one that strictly complies with the specification, requirements can be grouped into dimensions for which parameters are specified in order to evaluate the solution and measure the distance between its state at a given moment and the ideal specification, effectively establishing the level in which it fulfills the requirements. This methodology is compatible with Myers' (2005) system testing definition, being "a form of higher-order testing that compares the system or program to the original objectives" which requires "a written set of measurable objectives".

The following sub-chapters focus on the dimensions which are considered to be most relevant for the development process of this study's solution, with tests to be conducted to its functional, performance, security and compliance aspects.

7.1.1 Functional testing

Functional testing is a quality assurance process that bases its test cases on the specifications of the software component being scrutinized. In Myer's (2005) definition, "function testing is a process of attempting to find discrepancies between the program and the external specification". This testing procedure is classified as black box testing since internal program structure is rarely considered, with functions being tested based on the analysis of their output given certain inputs.

Despite what the nomenclature appears to indicate, this testing methodology is passible of being applied to modules and classes, and ultimately evaluates the compliance of a component with the specified functional requirements.

Sharing similarities with end-to-end testing, functional testing can be conducted on non-production environments, and is therefore more suitable for the established evaluation methodology, which allows us to assess the quality of the system at any given time.

7.1.1.1 Indicators

The following table describes the requirements which are followed by the indicators in place to test their fulfilment.

Identifier	Summary	Indicator
FR-001	The game state is correctly interpreted.	Boolean
FR-002	Pot odds are correctly calculated.	Boolean
FR-003	Expected value is correctly calculated.	Boolean
FR-004	Equity is correctly calculated.	Boolean
FR-005	Ranges are correctly evaluated.	Boolean
FR-006	Hands are correctly evaluated.	Boolean
FR-007	Opponents are modelled.	Boolean
FR-008	Opposing players are assigned pre-defined strategies.	Boolean
FR-009	Game trees are able to be generated given root node.	Boolean
FR-010	The simulation of all possibilities in a game tree is correctly conducted.	Boolean
FR-011	When simulating a game tree, a lookup occurs to see if the given situation has already been calculated.	Boolean
FR-012	Simulations always converge to a solution.	Boolean
FR-013	Data is stored in the database regarding the agent's operation.	Boolean
FR-014	Basic player statistics is demonstrated on the user-interface.	Boolean
FR-015	Players are able to initiate matches against the autonomous agent.	Boolean

7.1.2 Performance testing

Performance testing is a quality assurance practice used to determine the measure in which a system performs in terms of responsiveness and stability under defined workloads. It is used to measure the quality attributes of the system described in the FURPS+ model in relation to the performance requirements: Speed, efficiency, capacity and scalability.

The nature of the solution this study aims to develop makes it inherently performance-dependent, with concessions having to be made in order to preserve computational resources, simplify the task and produce results in a timely manner, since it is used in an active competitive environment and a decision can only take a certain amount of time before the play is forfeited.

7.1.2.1 Indicators

The following table describes the requirements which are followed by the indicators in place to test their fulfilment.

Table 21 - Hypothesis and indicators for Performance testing

Identifier	Hypothesis	Indicator
NFR-004	The solution is able to provide a response within the established time limit.	Boolean
NFR-005	The solution maintains performance throughout separate consecutive games.	Boolean

7.1.3 Security testing

Software security testing is the process of attempting to devise test cases that subvert an information system’s security mechanisms which protect data and assure functionality, such as the operating system’s memory protection mechanism or a database management system’s data security mechanisms (Myers, 2005).

Security requirements often specify elements of authentication and authorization, availability, confidentiality, integrity and irrefutability.

7.1.3.1 Indicators

The following table describes the requirements which are followed by the indicators in place to test their fulfilment.

Table 22 - Hypothesis and indicators for Security testing

Identifier	Hypothesis	Indicator
NFR-002	The client-server socket communication is secured via a TLS certificate.	Boolean
NFR-003	The database connection is performed securely with parameter sanitization.	Boolean
NFR-008	The solution is able to provide via a secure data transfer protocol.	Boolean

7.1.4 Compliance testing

Compliance testing is a form of non-functional testing which aims to validate a system's adherence to the stakeholder's prescribed standards, with the objective of determining if the solution's development and maintenance process meets the defined methodology, among other factors such as documentation.

7.1.4.1 Indicators

The following table describes the requirements which are followed by the indicators in place to test their fulfilment.

Table 23 - Hypothesis and indicators for Compliance testing

Identifier	Hypothesis	Indicator
NFR-001	The API provided adhered to the REST specification.	Boolean
NFR-006	The system's usability was created with ISO 9241's directives in mind.	Boolean
NFR-007	The solution's interface adheres to the specification.	Boolean
NFR-008	The solution is able to provide via a secure data transfer protocol.	Boolean
NFR-009	The server outputs JSON formatted data.	Boolean

7.2 Evaluation methodology

Due to the nature of the nature of the solution developed for the purposes of this study, a specific evaluation methodology is defined for its functionality and usability dimensions.

7.2.1 Functionality

The solution's overall functionality shall be demonstrated through the exhaustive analysis of the following critical qualities:

- **Computational performance:** The solution should be able to calculate and output a decision based on provided input within an established time interval of 15 seconds. This value is based on the average allotted time period allowed by internet Poker rooms, such as PokerStars.com, which provides 10 to 18 seconds depending on the type of game (*Pokerstars | Time bank in tournaments and cash games*, 2020).

- **In-game performance:** The solution's in-game performance should be demonstrated to be at least break-even, beating rake over a sample that is sizeable enough to overcome variance – the probability of a player's results not being due to variance can be measured using calculator software developed for this specific purpose (*Poker Variance Calculator - Primedope*, 2020).

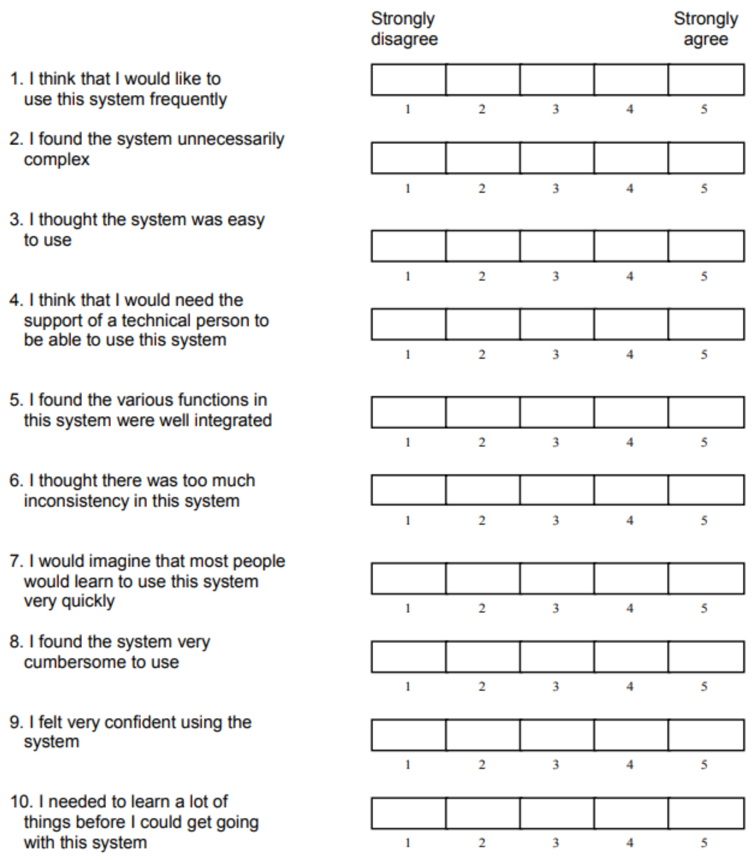
To ensure the existence of the aforementioned qualities, as well as the fulfilment of the remaining requirements, the solution will be subjected to unit and integration tests throughout its development process.

The definition of milestones along the development process where the system's efficiency is assessed provides a measurement of how the additions and changes implemented impact progress towards the final product's specification. With the definition and documentation of the stage of development in which each measurement was executed, a plain overview of how different features impact the system's efficiency in terms of computational and in-game performance will be available at the end of the development process, as well as a measure of how the system performs in relation to its state of the art counterparts described in chapter 0.

7.2.2 Usability

The user-interface allowing users to immerse in games facing the developed agent is an important portion of this solution which will be evaluated according to the System Usability Scale (SUS), a tool originally created by John Brooke in 1986 (Patrick W. Jordan, B. Thomas, Ian Lyall McClelland, 1996) which became an industry standard for the business and technology industries, providing a simple yet effective evaluation framework for a variety of services, in which websites and applications are included.

This scale consists of 10 items which respondents rate according to a Likert scale with five options (*System Usability Scale (SUS) | Usability.gov*, no date).



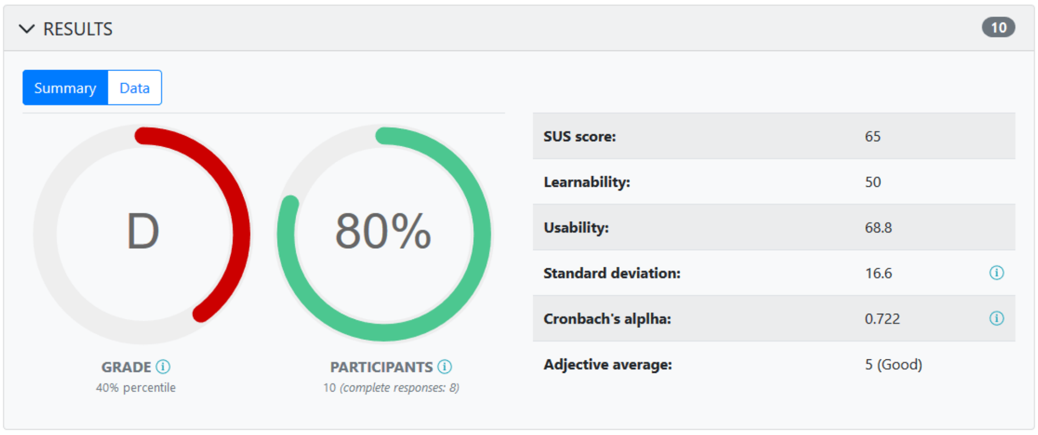
	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Figure 38 – System Usability Scale

These items measure different usability aspects such as:

- **Effectiveness:** Can users achieve their objectives;
- **Efficiency:** How effortlessly can users achieve their objectives;
- **Satisfaction:** To which degree are users satisfied when achieving their objectives.

For interpretation of the obtained results, each item's contribution is summed, regarding the item-specific method of assessing it – for items 1,3,5,7,9 the contribution is the scale position minus 1, and for the remaining, the contribution is 5 minus the scale position. Posteriorly to executing this assessment, scores are added and multiplied by 2.5, converting the original 0 to 40 scores to a 0 to 100 scale.


RESULTS 10

Summary Data

D

GRADE
40% percentile

80%

PARTICIPANTS
10 (complete responses: 8)

SUS score:	65
Learnability:	50
Usability:	68.8
Standard deviation:	16.6
Cronbach's alplha:	0.722
Adjective average:	5 (Good)

Figure 39 - Screenshot of usabiliTEST's results summary for System Usability Scale project (*System Usability Scale online with analytics | usabiliTEST*, 2020)

An adjective scale may be implemented, with studies demonstrating a high correlation between itself and the overall participant's description of their experience with a system (Bangor *et al.*, 2009), and tools such as System Usability Scale Plus[33] implementing this in the format of an enhanced SUS.

An evaluation in terms of usability shall be conducted on development milestones that demand it, such as the finalization of the initial user-interface prototype, evaluating it according to the aforementioned aspects and making improvements based on user feedback. The inquiries shall be conducted on a sufficiently significant sample of participants, with results being interpreted and documented in order to demonstrate how changes affected usability.

7.3 Evaluation results

In order to evaluate the developed solution, an in-depth analysis was performed to it's various components, assessing the degree to which the requirements were met in multiple dimensions. This chapter presents the results of the experimentation the solution was subjected to in terms of computational performance, in-game results, and usability.

7.3.1 Computational performance

Measuring the solution's computational performance is key to evaluating it's viability for many of the use-cases described at length in chapter 4.1.1.4, the object to which this study and it's applications are directed, is a game with a strict and succinct time limits for each

[33] usabilitiTEST's offers "SUS Plus", which offers 2 additonal options: Adjective rating scale and Promoter rating scale. More information on https://www.usabilitest.com/system-usability-scale.

decision. In order to assess the viability of such applications to real-world scenarios, the time elapsed in order to return information for each decision was studied.

Due to the nature of the solution's architecture, a significant processing overhead resides on the calculation component, due to which multiple measurements were taken in an attempt to segregate the various components of the application and perform this evaluation on an individual basis previously to performing it on the aggregate. This methodology provides insights on which components lack performance and require improvement, and which achieve an acceptable and constant performance figure.

All of the following tests were performed on a controlled environment with both the client and the server on the localhost. This eliminates variation caused by the differences in the server and client machine's geographical positions, as well as by the load of the server and internet-server-provider at the time of testing. While this improves the consistency of the test results, it should be noted that the performance benchmark discussed in this section is not to be expected in an online production environment.

7.3.1.1 Server

This analysis was started on the solution's entry point, the socket server which handles the client-server communication, the state of the game and passes it to the calculation component in order to obtain a game-play response. For the purposes of this test, this component was decoupled from the calculation component by changing it's algorithm to retrieve a random response within the universe of possibilities available for the current state of the game, instead of contacting the calculation component and listening for a response.

With industry-standard tools such as Postman lacking support for testing socket connections at the time of writing, and this being a highly specific test case, a script was developed in order to test the solution for this specific purpose. Developed using NodeJS and Socket.IO's Client API, this script connects to the server, starts a new game, and send an 'action' event to the server whenever it receives an 'act' event. The action is picked randomly and the time elapsed between sending the message and receiving the server's reply is the data intended to be collected in the course of this exercise.

Figure 40 chart's the response time of the server over 10 000 requests made by a mock adversary which also employed a mixed strategy, assuring a large sample or entropic game scenarios were tested. As can be observed, the performance remains relatively consistent throughout the sample, with an average response time of 25.95 ms, and a mean of 24 ms.

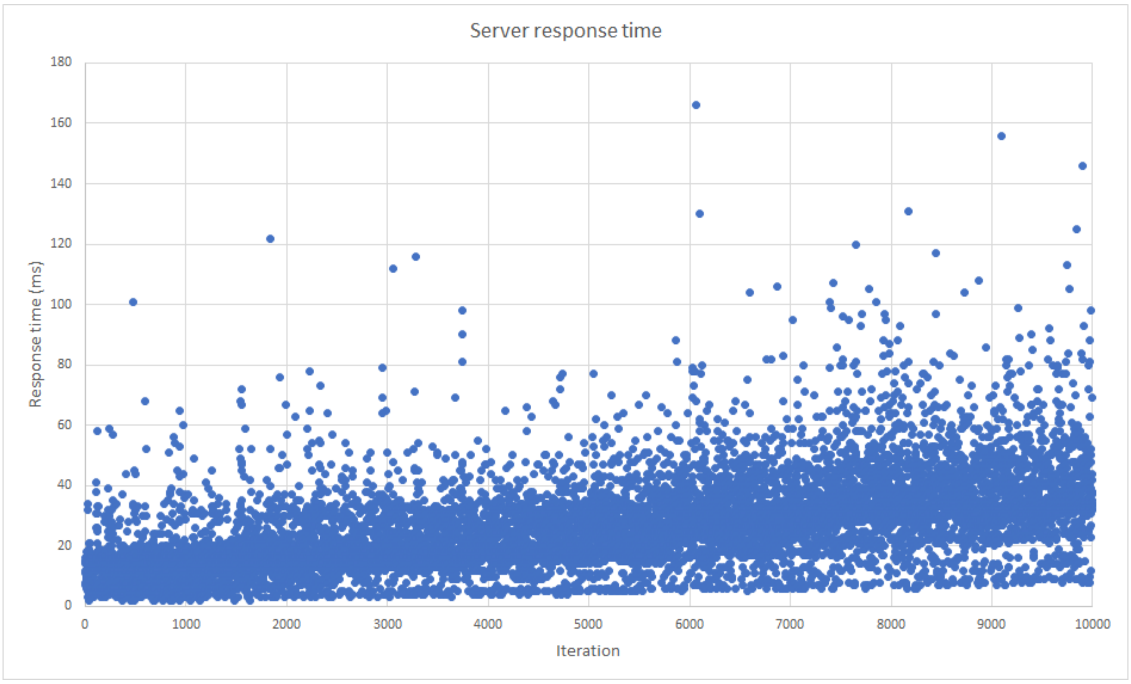


Figure 40 - Server response time

Additional statistics regarding Figure 40 can be seen in Table 24.

Table 24 - Server performance testing statistics

Statistic	Value
Average	25.95 ms
Mean	24 ms
Max	166.00 ms
Min	2.00 ms
Below average	55 %
Above average	45 %

The outlying values represent hands which reach showdown, forcing the solution to calculate the winning player according to the game's state. The equity calculation algorithm is used for this purpose for the sake of practicality and standardization. This isn't the most optimal decision in terms of performance, leaving room for improvement in this scenario.

It is also possible to notice a slight but consistent performance degradation, with the third quadrant's values being more concentrated on the lower part of the Y-axis in comparison to those in the fourth quadrant. A 100 000 iteration test was performed in order to see that this degradation was not proportional to the number of iterations. It was verified that this variation stagnates throughout the sample.

7.3.1.2 Calculator

Furthermore, the performance of the strategy calculation component was tested in a manner similar to that applied to the server component. It was invoked directly via a shell script which entropically conjured game states, ensuring once more that a thorough and conclusive sample can be obtained.

Figure 41 chart's the components response time over 10 000 calculations.

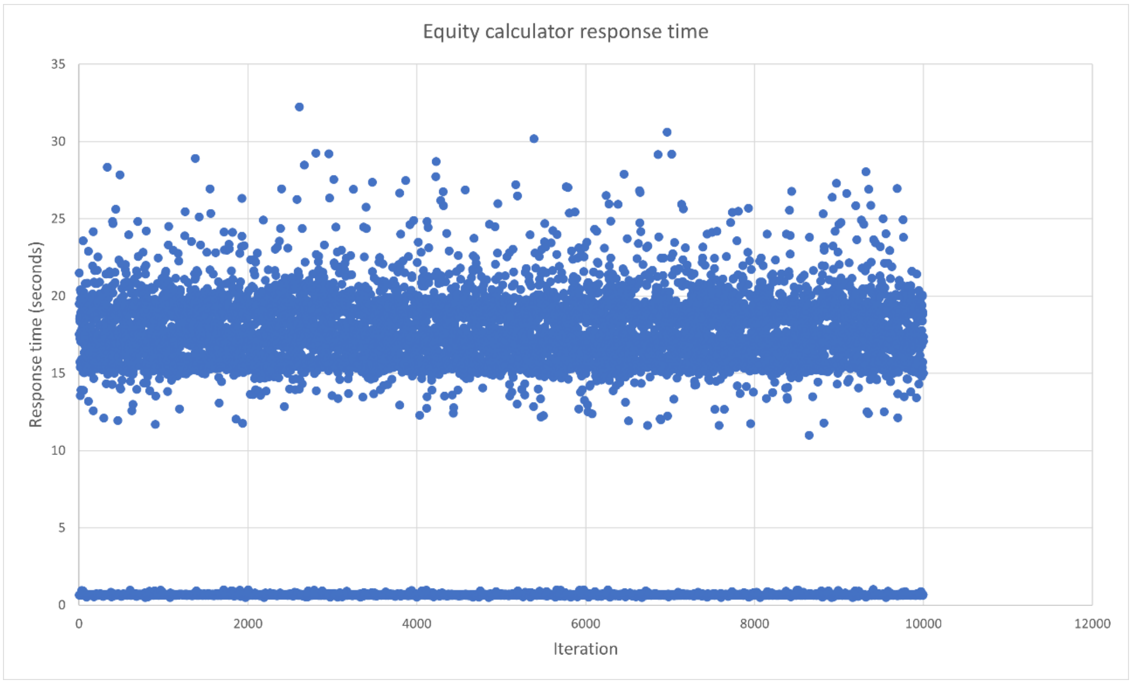


Figure 41 - Equity calculator response time

It is visible that the results are highly disperse. Further analysis was deemed necessary in order to dissect the scenarios in which performance varied. From a purely theoretical standpoint in which a relation of direct proportion can be established between the sample size of a game's state and the computation time needed to solve it, the dispersion in the results shown is justifiable due to the algorithm taking more time to compute the earlier stages of the game, where there is a larger amount of unknown community cards, directly affecting the equity calculation's performance. In this case, calculating the preflop stage of the game would take longer than in the flop, which in turn would take longer than in the turn and river stages. Figure 42, Figure 43, Figure 44 and Figure 45 show individual performance measurements in order to assess this theory.

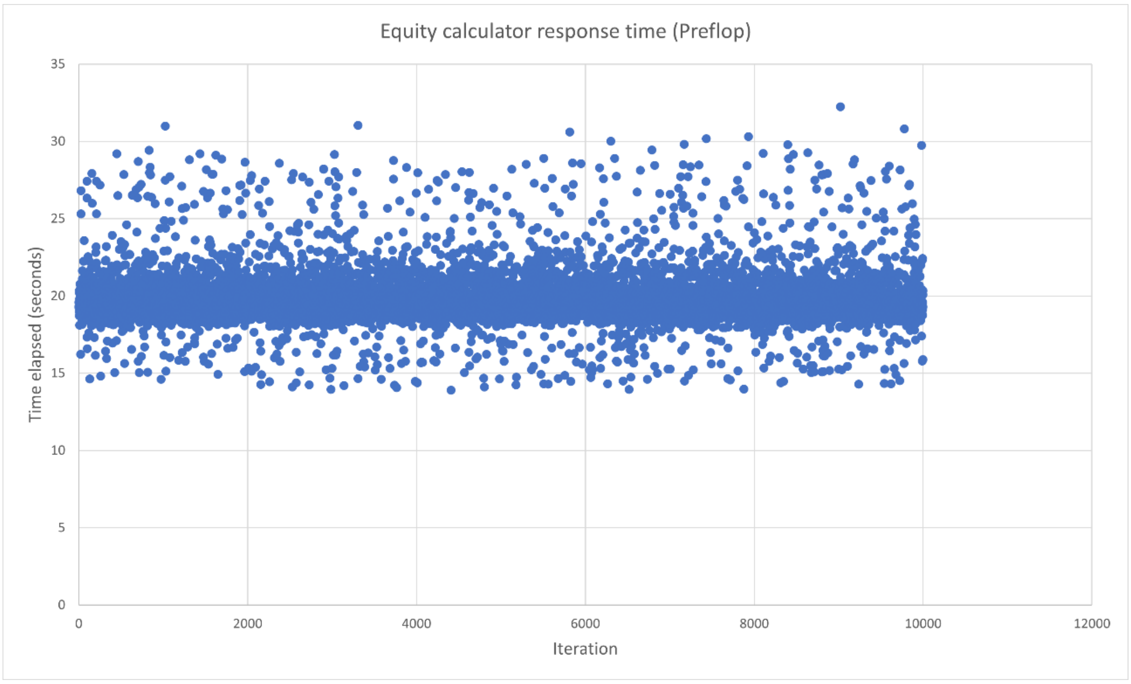


Figure 42 - Equity calculator response time in the preflop stage of the game

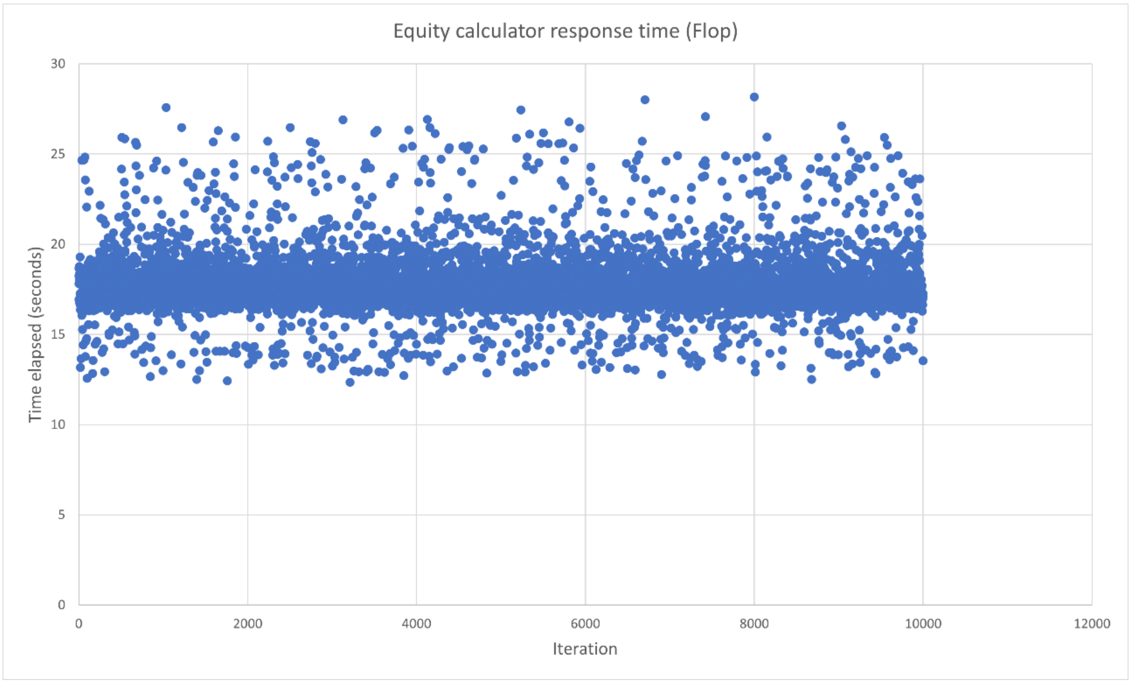


Figure 43 - Equity calculator response time in the flop stage of the game

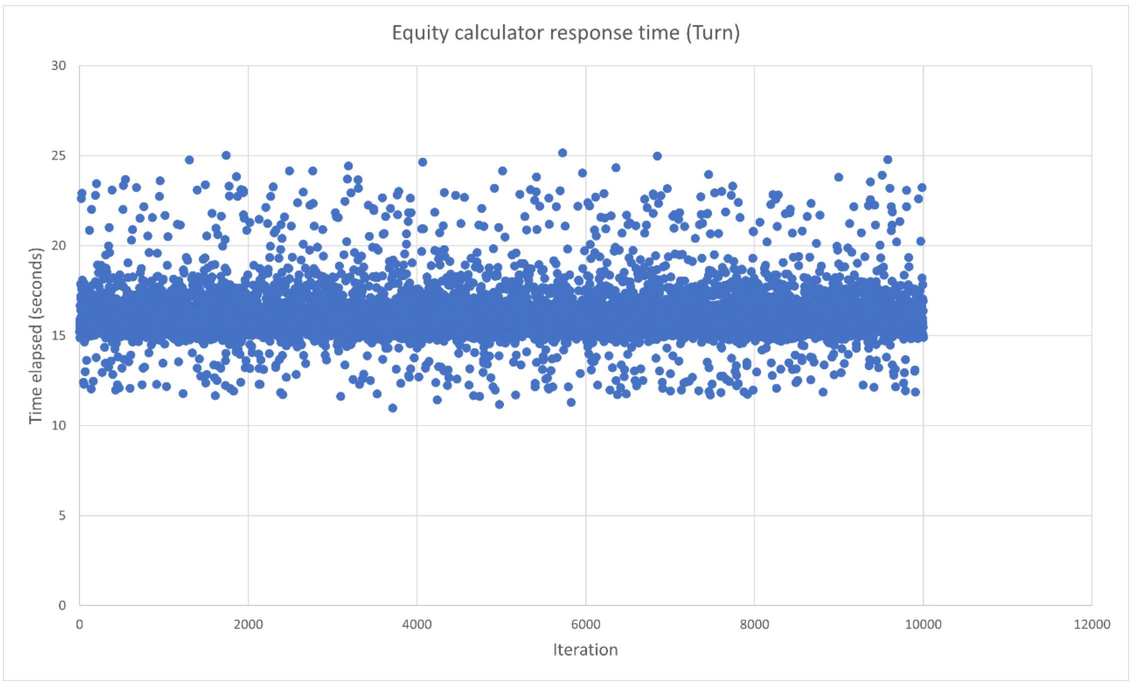


Figure 44 - Equity calculator response time in the turn stage of the game

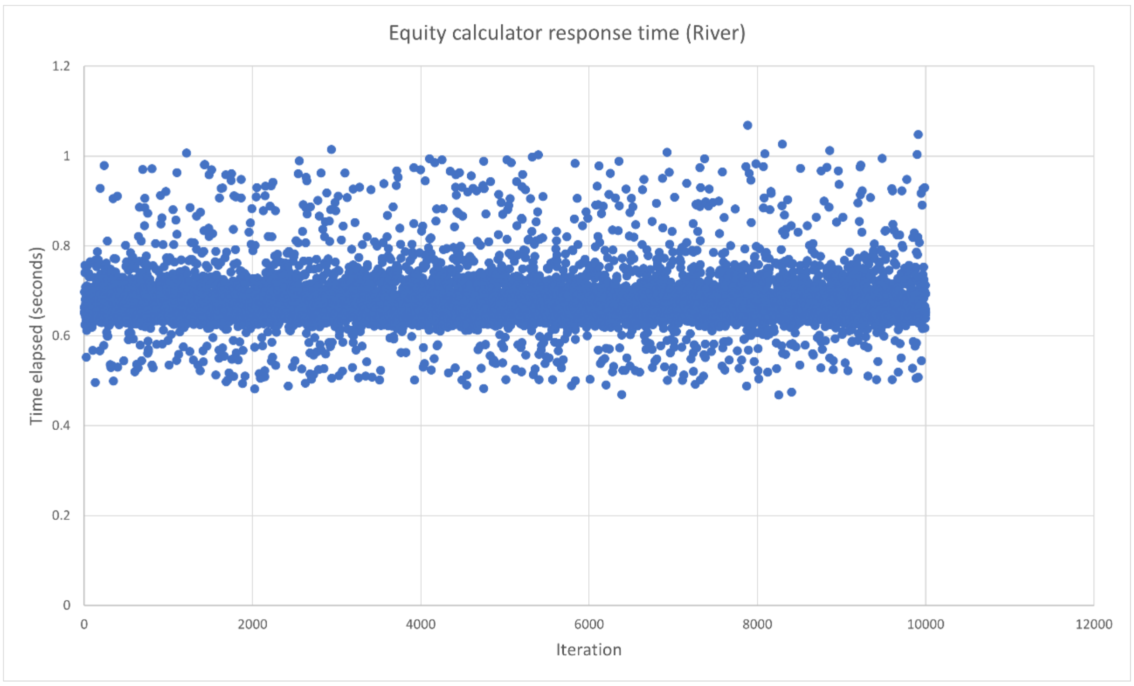


Figure 45 - Equity calculator response time in the river of the game

Statistics regarding the previous charts can be studied in Table 25.

Table 25 - Calculator performance testing statistics

Statistic	Value (seconds)
Average response time in the Preflop	19.67
Average response time in the Flop	17.59
Average response time in the Turn	15.99
Average response time in the River	0.67

Analyzing the figures above shows that the theoretical proposition is indeed correct, with a faster calculation the closer to the end of the game the state being calculated is.

One important factor to note is the results dispersed throughout the bottom of the vertical scale of Figure 41, largely separated from the remaining results. Paying attention to the scale of the pictures that follow, it is visible that the river calculations are far quicker than those on the remaining streets of the game. No additional community cards will be made available, which provides the calculator with more information, which lowers the sample size of the simulation to take place. Referring to chapter 6.1.2, the behavior observed here is due to the fact that the calculator takes a zealous approach to the problem in hand, enumerating the 42 570 possible outcomes of the hand available and calculating for each of them, instead of running 1 000 000 iterations of the Monte Carlo simulation.

7.3.1.3 Overall solution

Not less important is the macro view demonstrating the performance of the overall solution, which was tested in a manner similar to that described in the chapter 7.3.1.1. It is important to note that the database was truncated before the solution was submitted to testing, meaning every given game scenario was calculated in real time.

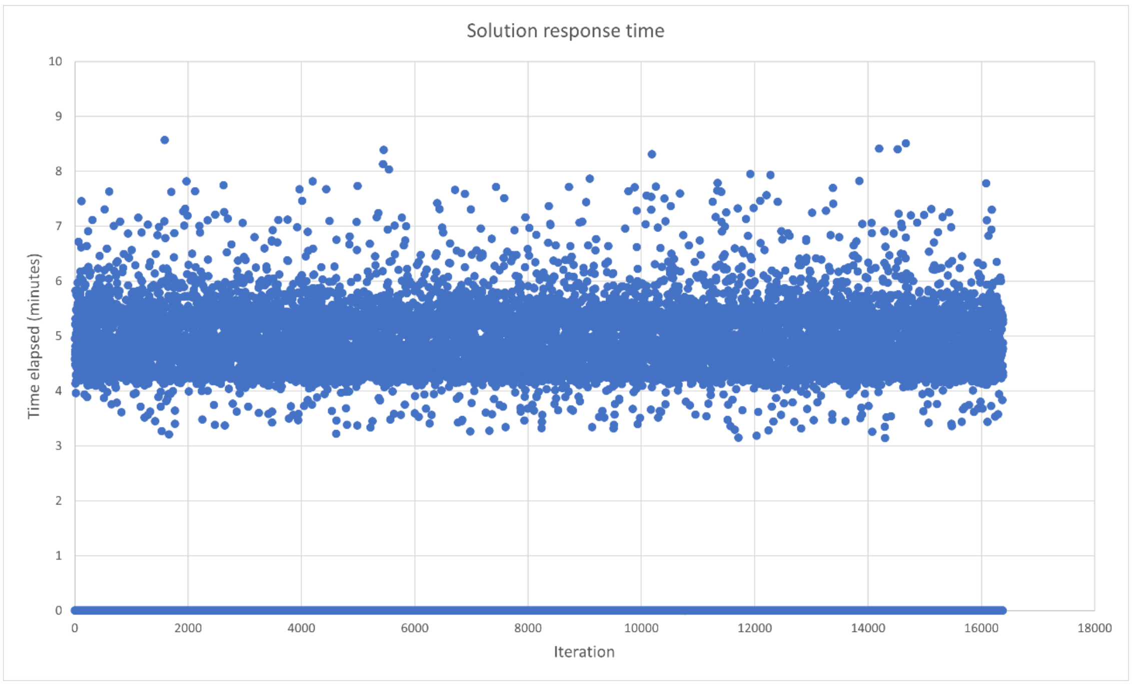


Figure 46 - Performance of the overall solution

As can be observed in Figure 46 - Performance of the overall solutionFigure 46, the solution does not meet the maximum response time requirements set, and therefor has reduced viability for usage in a real-time game-play scenario in it's current state.

Having already studied each component individually, it is clear that the performance optimization of the calculation component must be improved significantly were this the application to be given to this solution. Theoretically, a trade-off between processing and data storage can be performed, pre-calculating a large number of the more complex and time-requiring solutions and having them readily available for consultation in the application's database. However, the sample space of the game of No Limit Texas Hold'em potentially renders this option unfeasible.

7.3.2 In-game performance

An highly capable solution in terms of computational performance is hindered if the insights it provides are of little to no use. More than performing quickly, the objective of this solution is to provide highly accurate game-theoretically optimal solutions which opponents are mathematically incapable of exploiting.

In order to assess the solution in terms of it's in-game capabilities, strategy profiles with various characteristics were play-tested against it. The following strategy profiles were considered:

- **Strategy profile A:** Pure strategy where the agent always folds. This acts as a baseline;
- **Strategy profile B:** Pure strategy where the agent always raises;
- **Strategy profile C:** Totally mixed strategy where the agent picks actions at random (but never folds when there is the possibility to check);
- **Strategy profile D:** The same strategy as the solution;

These strategies were tested versus the solution over a sample of 10 000 hands in the simplified game of shove-fold[34] in order to ease and accelerate the testing procedure. The significance of the sample is of great relevance in order to exclude the variance factor from the results as much as possible.

Figure 47 display the solution's performance over the described sample.

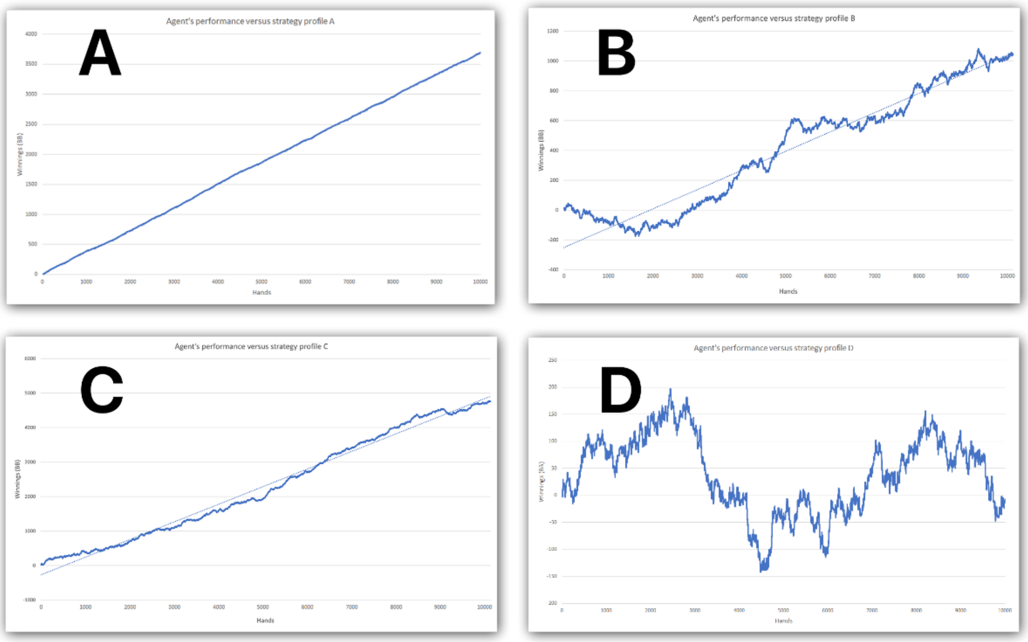
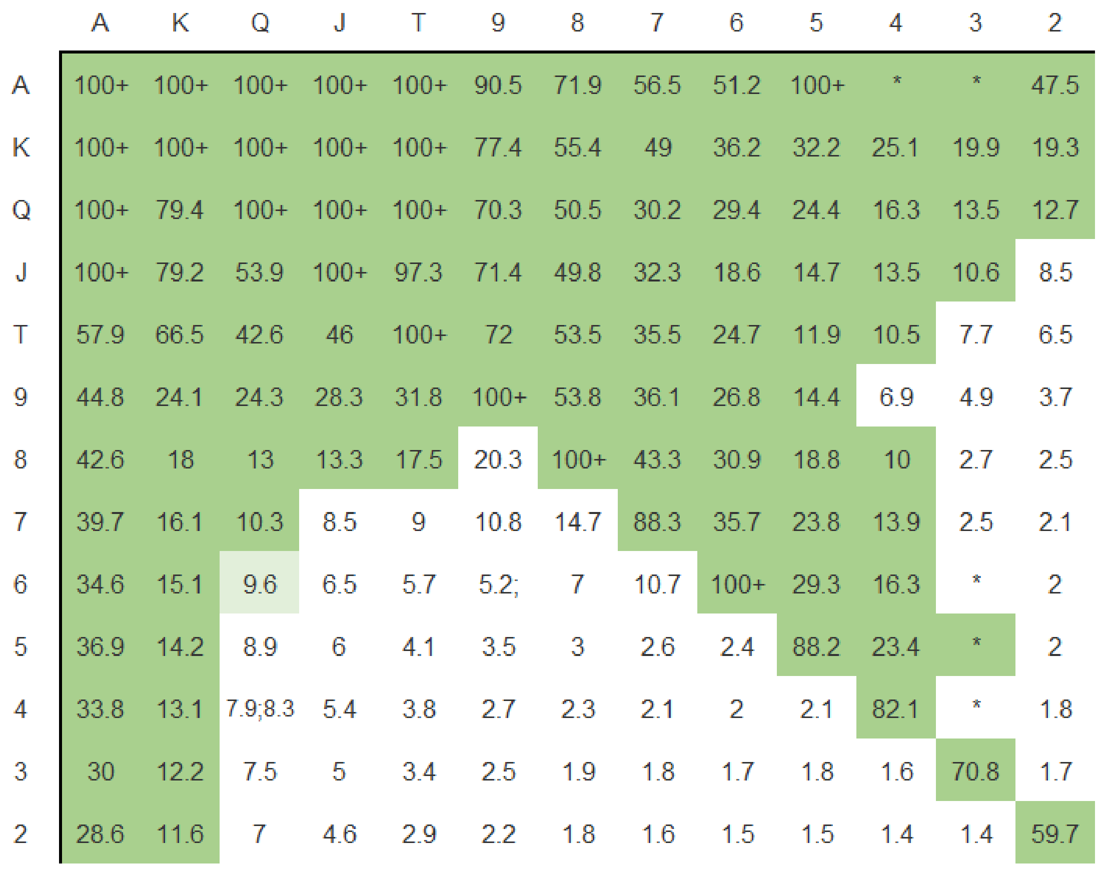


Figure 47 - Solution's in-game performance

The figures demonstrate that the solution is a clear winner versus adversaries employing strategies such as the ones described. Furthermore, the agent's performance record versus strategy profile D indicates that this strategy approximates the Nash equilibrium, as the expectation of playing against a mirror opponent over a significant sample is null.

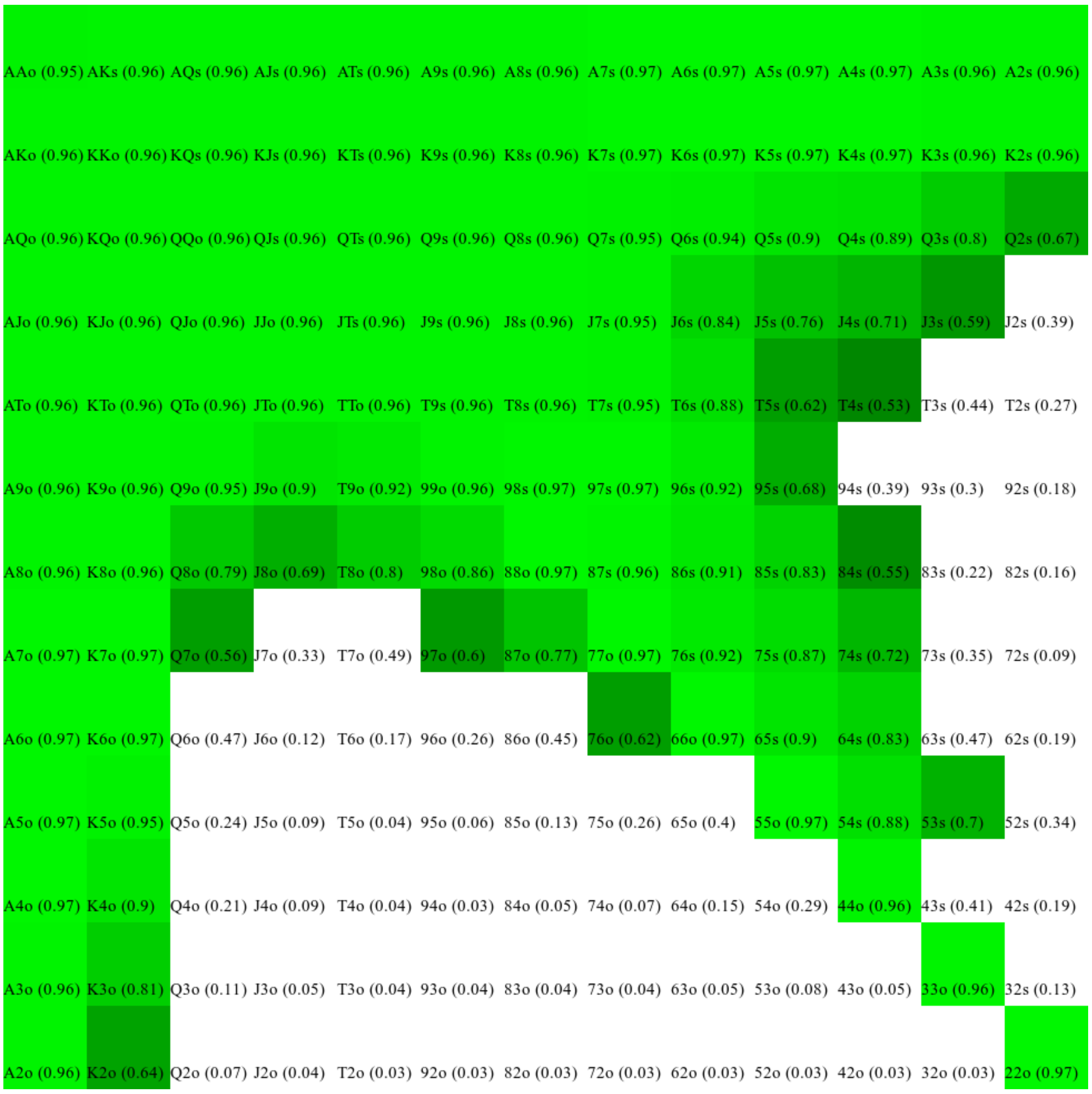
[34] Simplified game where players with equal chip stacks can only opt to raise or call an all-in bet or fold.

Testing the application from a theoretical standpoint is also possible by computing Nash equilibrium charts and comparing them with officially recognized solutions which can be consulted in books such as Expert Heads Up No Limit Hold'em: Optimal And Exploitative Strategies (Poker Series) Volume 1: Optimal and Exploitative Strategies (Tipton, 2012).



	A	K	Q	J	T	9	8	7	6	5	4	3	2
A	100+	100+	100+	100+	100+	90.5	71.9	56.5	51.2	100+	*	*	47.5
K	100+	100+	100+	100+	100+	77.4	55.4	49	36.2	32.2	25.1	19.9	19.3
Q	100+	79.4	100+	100+	100+	70.3	50.5	30.2	29.4	24.4	16.3	13.5	12.7
J	100+	79.2	53.9	100+	97.3	71.4	49.8	32.3	18.6	14.7	13.5	10.6	8.5
T	57.9	66.5	42.6	46	100+	72	53.5	35.5	24.7	11.9	10.5	7.7	6.5
9	44.8	24.1	24.3	28.3	31.8	100+	53.8	36.1	26.8	14.4	6.9	4.9	3.7
8	42.6	18	13	13.3	17.5	20.3	100+	43.3	30.9	18.8	10	2.7	2.5
7	39.7	16.1	10.3	8.5	9	10.8	14.7	88.3	35.7	23.8	13.9	2.5	2.1
6	34.6	15.1	9.6	6.5	5.7	5.2;	7	10.7	100+	29.3	16.3	*	2
5	36.9	14.2	8.9	6	4.1	3.5	3	2.6	2.4	88.2	23.4	*	2
4	33.8	13.1	7.9;8.3	5.4	3.8	2.7	2.3	2.1	2	2.1	82.1	*	1.8
3	30	12.2	7.5	5	3.4	2.5	1.9	1.8	1.7	1.8	1.6	70.8	1.7
2	28.6	11.6	7	4.6	2.9	2.2	1.8	1.6	1.5	1.5	1.4	1.4	59.7

Figure 48 - Heads Up No Limit Hold'em shove/fold game equilibrium for the SB



AAo (0.95)	AKs (0.96)	AQs (0.96)	AJs (0.96)	ATs (0.96)	A9s (0.96)	A8s (0.96)	A7s (0.97)	A6s (0.97)	A5s (0.97)	A4s (0.97)	A3s (0.96)	A2s (0.96)
AKo (0.96)	KKo (0.96)	KQs (0.96)	KJs (0.96)	KTs (0.96)	K9s (0.96)	K8s (0.96)	K7s (0.97)	K6s (0.97)	K5s (0.97)	K4s (0.97)	K3s (0.96)	K2s (0.96)
AQo (0.96)	KQo (0.96)	QQo (0.96)	QJs (0.96)	QTs (0.96)	Q9s (0.96)	Q8s (0.96)	Q7s (0.95)	Q6s (0.94)	Q5s (0.9)	Q4s (0.89)	Q3s (0.8)	Q2s (0.67)
AJo (0.96)	KJo (0.96)	QJo (0.96)	JJo (0.96)	JTs (0.96)	J9s (0.96)	J8s (0.96)	J7s (0.95)	J6s (0.84)	J5s (0.76)	J4s (0.71)	J3s (0.59)	J2s (0.39)
ATo (0.96)	KTo (0.96)	QTo (0.96)	JTo (0.96)	TTo (0.96)	T9s (0.96)	T8s (0.96)	T7s (0.95)	T6s (0.88)	T5s (0.62)	T4s (0.53)	T3s (0.44)	T2s (0.27)
A9o (0.96)	K9o (0.96)	Q9o (0.95)	J9o (0.9)	T9o (0.92)	99o (0.96)	98s (0.97)	97s (0.97)	96s (0.92)	95s (0.68)	94s (0.39)	93s (0.3)	92s (0.18)
A8o (0.96)	K8o (0.96)	Q8o (0.79)	J8o (0.69)	T8o (0.8)	98o (0.86)	88o (0.97)	87s (0.96)	86s (0.91)	85s (0.83)	84s (0.55)	83s (0.22)	82s (0.16)
A7o (0.97)	K7o (0.97)	Q7o (0.56)	J7o (0.33)	T7o (0.49)	97o (0.6)	87o (0.77)	77o (0.97)	76s (0.92)	75s (0.87)	74s (0.72)	73s (0.35)	72s (0.09)
A6o (0.97)	K6o (0.97)	Q6o (0.47)	J6o (0.12)	T6o (0.17)	96o (0.26)	86o (0.45)	76o (0.62)	66o (0.97)	65s (0.9)	64s (0.83)	63s (0.47)	62s (0.19)
A5o (0.97)	K5o (0.95)	Q5o (0.24)	J5o (0.09)	T5o (0.04)	95o (0.06)	85o (0.13)	75o (0.26)	65o (0.4)	55o (0.97)	54s (0.88)	53s (0.7)	52s (0.34)
A4o (0.97)	K4o (0.9)	Q4o (0.21)	J4o (0.09)	T4o (0.04)	94o (0.03)	84o (0.05)	74o (0.07)	64o (0.15)	54o (0.29)	44o (0.96)	43s (0.41)	42s (0.19)
A3o (0.96)	K3o (0.81)	Q3o (0.11)	J3o (0.05)	T3o (0.04)	93o (0.04)	83o (0.04)	73o (0.04)	63o (0.05)	53o (0.08)	43o (0.05)	33o (0.96)	32s (0.13)
A2o (0.96)	K2o (0.64)	Q2o (0.07)	J2o (0.04)	T2o (0.03)	92o (0.03)	82o (0.03)	72o (0.03)	62o (0.03)	52o (0.03)	42o (0.03)	32o (0.03)	22o (0.97)

Figure 49 - Heads Up No Limit Hold'em shove/fold game equilibrium for the SB considering an effective stack size of 10 BB

Figure 48 demonstrates the Nash equilibrium strategy in a shove/fold heads up game, with each hand's value representing the stack size up to which player's shove, while Figure 49 demonstrates the solution obtained by the solution's algorithm for the same game. The highlighted hands those with which players shove with a stack size of up to 10 big-blinds. It is shown that both solutions collide perfectly, proving the effectiveness of the solution's strategy calculation algorithm.

7.3.3 Usability

With the end-user interacting directly with the product developed, a study was undertaken with the objective of assessing the degree to which it fulfills the usability guidelines specified by standards such as ISO 9241. In this chapter, the methodology, execution procedure and results of this study are presented.

As stated in chapter 7.2.2, the methodology documented by the System Usability Scale was used in order to conduct this survey and draw conclusions. Thus, the body of the survey consisted of SUS' standard questions, which can be consulted in Figure 38. Google Forms was used in order to ease the survey's distribution and the analysis of it's results.

This study consisted of a survey directed at a general demographic in order to obtain an unbiased set of results from which an evaluation onto the user interface's plus and minus points can be drawn. Individuals with ages ranging from 19 to 55 were interviewed in a setting where they were able to calmly utilize the application. Due to the world's social distancing context at the time of writing, all of the interviews were mediated remotely. A total of 14 answers were gathered.

The results are demonstrated in Table 26, with each question's the most voted option being highlighted.

Table 26 - Usability survey results

Question	1	2	3	4	5
1. I think that I would like to use this system frequently.	0 (0%)	0 (0%)	0 (0%)	3 (21%)	**11 (79%)**
2. I found the system unnecessarily complex.	**11 (79%)**	3 (21%)	0 (0%)	0 (0%)	0 (0%)
3. I thought the system was easy to use.	0 (0%)	0 (0%)	0 (0%)	3 (21%)	**11 (79%)**
4. I think that I would need the support of a technical person to be able to use this system.	0 (0%)	0 (0%)	0 (0%)	**12 (86%)**	2 (14%)
5. I found the various functions in this system were well integrated.	0 (0%)	0 (0%)	1 (7%)	4 (29%)	**9 (64%)**
6. I thought there was too much inconsistency in the system.	0 (0%)	0 (0%)	0 (0%)	**12 (86%)**	2 (14%)
7. I would imagine that most people would learn to use this system very quickly.	0 (0%)	0 (0%)	0 (0%)	4 (29%)	**10 (71%)**
8. I found the system very cumbersome to use.	**10 (71%)**	4 (29%)	0 (0%)	0 (0%)	0 (0%)
9. I felt very confident using the system.	0 (0%)	0 (0%)	2 (14%)	3 (21%)	**9 (64%)**
10. I needed to learn a lot of things before I could get going with this system.	0 (0%)	0 (0%)	**10 (71%)**	3 (21%)	1 (7%)

The treatment and data analysis of the data is imposed by the methodology in use and is documented in chapter 7.2.2. The results were then compared with the methodology's reference values, shown in Table 27.

Table 27 - Reference adjective values for SUS score

Score	Grade	Adjective classification
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Acceptable
51 - 68	D	Poor
< 51	F	Worst imaginable

It is important to note that even though the values are presented within a 0 to 100 scale, they do not reflect a percentual value, and should therefore be interpreted based on the reference adjective values.

An overall breakdown of the results is visible in Table 28.

Table 28 - Breakdown of the SUS results

Interviewee	Score	Grade
1	97.5	A
2	100	A
3	87.5	A
4	87.5	A
5	97.5	A
6	95	A
7	90	A
8	97.5	A
9	100	A
10	82.5	A
11	90	A
12	100	A
13	77.5	B
14	100	A
Average	92.08	A

Analyzing the results, it is verifiable that the usability is satisfactory as all the questionnaire scores exceed 77.5, a figure associated with an adjective classification of "Good".

There are pitfalls associated with the particular methodology used in order to assess the usability of this application. Since this is, on top of the complexity of the underlying elements, a card game, most people are already quite familiarized with the concept. The elements that the application consists of also lack complexity from a usability standpoint. For these reasons, the survey was geared towards evaluating how consistent, performant and visually up-to-standards the application is, instead of it's overall usability as is the case in regular, non-gaming applications.

7.3.4 Requirements

Given the aforementioned test results of it's computational, in-game and usability performance, the set of requirements put in place for the current solution was evaluated.

Starting with the functional requirements, Table 29 demonstrates that 10 out of 15 requirements were fulfilled, meaning that 66 % of the functional requirements set were implemented and functional in the final state of the solution.

Table 29 - Functional requirements evaluation

Requirement	Description	Evaluation
FR-001	The game's state is correctly interpreted.	1
FR-002	Pot odds are correctly calculated.	0
FR-003	Expected value is correctly calculated.	1
FR-004	Equity is correctly calculated.	1
FR-005	Ranges are correctly evaluated.	1
FR-006	Hands are correctly evaluated.	1
FR-007	Opponents are modelled.	1
FR-008	Opposing players are assigned pre-defined strategies	1
FR-009	Game trees are able to be generated given root node.	0
FR-010	The simulation of all possibilities in a game tree is correctly conducted.	0
FR-011	When simulating a game tree, a lookup occurs to see if the given situation has already been calculated.	0
FR-012	Simulations always converge to a solution.	1
FR-013	Data is stored in the database regarding the agent's operation.	1
FR-014	Basic player statistics is demonstrated on the user-interface.	0
FR-015	Players are able to initiate matches against the autonomous agent.	1

Proceeding with the non-functional requirements, Table 30 shows that 7 requirements have been met out of the total 9, for a ratio of 78 %.

Table 30 - Non-functional requirements evaluation

Requirement	Description	Evaluation
NFR-001	The API provided adhered to the REST specification.	0
NFR-002	The client-server socket communication is secured via a TLS certificate.	1
NFR-003	The database connection is performed securely with parameter sanitization	1
NFR-004	The solution is able to provide a response within the established time limit.	0
NFR-005	The solution maintains performance throughout separate consecutive games.	1
NFR-006	The system's usability was created with ISO 9241's directives in mind.	1
NFR-007	The solution's interface adheres to the specification.	1
NFR-008	The solution is able to provide via a secure data transfer protocol.	1
NFR-009	The server outputs JSON formatted data.	1

Joining the completion ratios of both the functional and non-functional requirements, the application fulfills 71% of the requirements which were initially set – a figure which isn't poor considering that the decision of implementing the fictitious play algorithm instead of counterfactual regret minimization causes the solution to lack a game tree, preventing it from fulfilling 3 of the functional requirements.

8 Conclusion

This chapter provides an analysis of what was achieved through the development of this project based on the objectives and the degree to which they were fulfilled. It summarizes the lessons, challenges, conquests and defeats which are a product of the development of this study and aims to capture what could be improved in future works.

8.1 Overview

Analysis of chapter 0 provides great insight on the success of the development process. The main goal of creating an autonomous Poker agent was achieved, formulating an algorithm capable of learning to play the Heads Up No Limit Texas Hold'em game to an optimal degree in theoretical terms, which assures that it won't be a losing agent.

While the aforementioned achievement is valuable for the number of valuable lessons it provides in terms of game-theory and machine learning concepts, a more sophisticated solution, using a state-of-the-art machine learning algorithm more commonly used in the break-through studies in this field, would be a better fit were this solution to be put to use in one of the scenarios described in 4.1.4.4. Due to the high processing capability or storage space necessities of the algorithm in use, it's scalability is jeopardized, and therefor so is it's real-time usage scenarios.

8.2 Limitations and improvements

The lack of performance and scalability of the solution is it's greatest flaw. With more concise knowledge regarding the performance necessities of such an application, it is evident that a better decision could have been made in regards to which language to base the autonomous agent's strategy evaluating algorithm on. While Python's native data accessibility and manipulation capabilities are invaluable, a language that is precompiled and supports multi-threading and/or multi-processing, such as C, would have allowed for a significantly improved computational performance. Would such an agent be intended for use in real-time play situations with a strict time limit imposed on each decision, improving the performance of the most resource and processor time intensive component of the solution is a must.

As was previously mentioned, a more modern and proven approach than the one applied in this project would lead to better results.

Several optimization methods which could potentially improve the performance of the application were also left to be tested and implemented – a great example of one is the calculation of the exploitability of the current solution, setting the stopping point of the algorithm to an exploitability threshold instead of executing it for a set of iterations.

Finally, more in-depth testing methodologies should be applied were this application to be launched into a real-world production environment. While testing the strategies calculated for simplified games such as the shove/fold is a practical approach which has it's merits and facilitates testing, tests on the full game should be conducted in order to assess factors such as which scenarios the solution might behave unexpectedly in, and the degree to which having a large set of pre-calculated solutions may improve performance.

8.3 Final thoughts

When dealing with algorithms that run millions of iterations composed of non-trivial procedures, it is vital to utilize every last bit of performance available. This project allowed me to better grasp the performance impacts of the decisions applied to the design and implementation of applications, understanding the cost each innocent looking operation might hold upon the execution of the entire algorithm.

Dealing with mathematical implementation of game theory principles is also a challenge I enjoyed and believe every individual with a passion for programming would evolve by experiencing it.

While I am satisfied with the outcome of the project, the solution which I would deem perfect is far from being achieved. I consider the final product to be in an acceptable state, and now realize that the initial expectations set for this project were far too ambitious.

In a time where the general public's awareness for real-time assistance (RTA) applications is rapidly growing in the world of online Poker, it is crucial for the game-provider networks to position themselves in the bleeding-edge of this field if they wish to preserve the vitality of the online Poker playing ecosystem, which is founded on a trust relation formed with their customers.

The investigation of topics such as the one the present study focuses on can potentially aid these entities in understanding the methodologies used by dishonest actors and lead to the discovery of new, more efficient ways of dealing with these situations.

As game theory has it, as long as there is a viable profitable trade-off, there will be actors pushing the boundaries of our knowledge in order to maximize their utility – and this applies not only to the game of Poker, but to all trade-off relationships we, as humans, may find in the game of life.

9 Appendix A – Poker glossary

3-bet	When there is a bet, raise and re-raise, the re-raise is called a "3-bet".
6-max	A table in which up to 6 players can participate.
9-max	A table in which up to 9 players can participate.
All-in	A bet for all of a player's chips.
Ante	Mandatory bet all players are forced to make, regardless of their position.
ATC	Acronym for "any two cards", typically meaning a random 2 card subset from the deck.
Blinds	Mandatory bet players in the small-blind and big-blind positions are forced to perform.
Blind defense	When the players in the blinds re-raise in order to defend a pot composed of the chips paid due to the blinds.
Blind stealing	When players raise in late-position in order to take the pot from the players in the small-blind and big-blind positions.
Blind-versus-blind	Duel between the small-blind and big-blind players.
Blockers	Cards which prevent adversaries from completing a hand.
Bluff	Betting without a strong hand.

Board	Refers to the community cards.
Broadway	Cards larger than 9.
Collusion	When two players exchange private information in order to increase their individual or combined utility.
Community cards	Cards which all players have access to, composing the flop, turn and river.
Dead money	Chips in the pot which no player has a particular incentive to protect, typically put there via ante or blind.
Draw	(flush or straight)… drawing.. open-ended…
Effective stack	The largest stack of all the players involved in a play.
Hand	Can refer to a play or a player's hole cards.
Heads-up	A game between two players.
Hole cards	Players' private cards.
Implied odds	Extension of pot odds which help decide if it is worth calling a raise with a drawing hand.
In-position	A player is said to be "in-position" when he acts after his opponent, knowing his decision beforehand.
Preflop	The stage of the game when no community card is yet visible.
Postflop	The flop, turn and river stages of the game.
Pot	The total amount of chips bet up to the given moment in the current hand.
Pot odds	The ratio between the size of the total pot and the size of the bet a player faces.
Offsuit	Cards of different suits.
Open	To place the first bet in a betting round.
Open-ended	A straight which can be completed by two cards.
Out of position	A player is said to be "out of position" when he acts first.
Overbet	A bet larger than the pot.
Shove	An all-in bet.

Stack	The amount of chips belonging to each player.
Street	Betting round (preflop, flop, turn and river).
Suited	Cards of the same suit.

10 Appendix B – Criteria's consistency

10.1 Implementation

$$Criteria\,matrix\ \times Eigenvector \cong\ \lambda_{max}\ \times Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & \frac{1}{3} & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 3 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.20 \\ 0.15 \\ 0.20 \\ 0.25 \\ 0.20 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.20 \\ 0.15 \\ 0.20 \\ 0.25 \\ 0.20 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 1 \\ 0.8333 \\ 1 \\ 1.3 \\ 1 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.20 \\ 0.15 \\ 0.20 \\ 0.25 \\ 0.20 \end{bmatrix}$$

$$\Leftrightarrow\ \lambda_{max} \cong 5.15$$

Knowing the λ_{max} value the CI and CR can now be calculated, considering $n = 5$ and $RI = 1{,}12$.

$$CI = \frac{\lambda_{max} - n}{n - 1} =\ 0.0375 \qquad\qquad CR = \frac{CI}{RI} = 0.03$$

10.2 Hardware requirements

$$Criteria\,matrix \times Eigenvector \cong \lambda_{max} \times Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 9 & 9 & 9 & 9 \\ \frac{1}{9} & 1 & 3 & 5 & 7 \\ \frac{1}{9} & \frac{1}{3} & 1 & 3 & 5 \\ \frac{1}{9} & \frac{1}{5} & \frac{1}{3} & 1 & 1 \\ \frac{1}{9} & \frac{1}{7} & \frac{1}{5} & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.60 \\ 0.18 \\ 0.10 \\ 0.08 \\ 0.04 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.60 \\ 0.18 \\ 0.10 \\ 0.08 \\ 0.04 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 7.4 \\ 3.42 \\ 2.02 \\ 0.57 \\ 0.534 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.60 \\ 0.18 \\ 0.10 \\ 0.08 \\ 0.04 \end{bmatrix}$$

$$\Leftrightarrow \lambda_{max} \cong 14.4$$

Knowing the λ_{max} value the CI and CR can now be calculated, considering $n = 5$ and $RI = 1{,}12$.

$$CI = \frac{\lambda_{max} - n}{n-1} = 2.35 \qquad CR = \frac{CI}{RI} = 2.09$$

10.3 Time elapsed

$$Criteria\,matrix \times Eigenvector \cong \lambda_{max} \times Eigenvector$$

$$\Leftrightarrow \begin{bmatrix} 1 & 9 & 9 & 9 & 9 \\ \frac{1}{9} & 1 & 3 & 5 & 7 \\ \frac{1}{9} & \frac{1}{3} & 1 & 3 & 5 \\ \frac{1}{9} & \frac{1}{5} & \frac{1}{3} & 1 & 1 \\ \frac{1}{9} & \frac{1}{7} & \frac{1}{5} & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0.60 \\ 0.18 \\ 0.10 \\ 0.08 \\ 0.04 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.60 \\ 0.18 \\ 0.10 \\ 0.08 \\ 0.04 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} 7.4 \\ 3.42 \\ 2.02 \\ 0.57 \\ 0.534 \end{bmatrix} \cong \lambda_{max} \times \begin{bmatrix} 0.60 \\ 0.18 \\ 0.10 \\ 0.08 \\ 0.04 \end{bmatrix}$$

$$\Leftrightarrow \lambda_{max} \cong 14.4$$

Knowing the λ_{max} value the CI and CR can now be calculated, considering $n = 5$ and $RI = 1{,}12$.

$$CI = \frac{\lambda_{max} - n}{n - 1} = 2.35 \qquad\qquad CR = \frac{CI}{RI} = 2.09$$

References

Alberta, U. of (2008) *The Second Man-Machine Poker Competition*. Available at: http://webdocs.cs.ualberta.ca/~games/poker/man-machine/ (Accessed: 3 September 2020).

Alex (2014) *Strategies, Nash Equilibria and GTO Poker*. Available at: http://blog.gtorangebuilder.com/2014/03/strategies-nash-equilibria-and-gto.html (Accessed: 5 August 2020).

Ambler, S. W. (no date) *User Stories: An Agile Introduction*. Available at: http://www.agilemodeling.com/artifacts/userStory.htm (Accessed: 22 February 2020).

Bangor, A. *et al.* (2009) 'Determining what individual SUS scores mean: adding an adjective rating scale', *Determining what individual SUS scores mean: adding an adjective rating scale*, 4(3), pp. 114–123.

Bartone, L. and While, L. (2000) 'Adaptive Learning for Poker', *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 560–573.

Van Benthem, J. and Ter Meulen, A. (2011) *Handbook of Logic and Language*, *Handbook of Logic and Language*. doi: 10.1016/C2010-0-65666-5.

Billings, D. *et al.* (1998) 'Opponent modeling in poker', *Proceedings of the National Conference on Artificial Intelligence*, pp. 493–499.

Bowling, M. *et al.* (2015) 'Heads-up limit hold'em poker is solved', *Science*, (09 Jan 2015), pp. 145–149. doi: 10.1126.

Brown, N. (2019) 'Facebook, Carnegie Mellon build first AI that beats pros in 6-player poker', *Facebook AI Research*, 7 November. Available at: https://ai.facebook.com/blog/pluribus-first-ai-to-beat-pros-in-6-player-poker/.

Calculating Win Odds for Multiple Opponents - Poker for Programmers: Poker Algorithms and Tools for the C# Programmer (2006). Available at: http://pokerforprogrammers.blogspot.com/ (Accessed: 9 February 2020).

Campos, J. (2013) 'A Profitable Online Poker Agent'.

Can I use... Web Sockets (2020). Available at: https://caniuse.com/websockets (Accessed: 4 October 2020).

Carnegie Mellon, U. (2017) 'Poker Play Begins in "Brains Vs. AI: Upping the Ante" - News - Carnegie Mellon University', January, p. 1. Available at: https://www.cmu.edu/news/stories/archives/2017/january/poker-play.html#:~:text=Play began Jan.,world's best professional poker players.

Chad Holloway, 888 Poker (2017) *The Difference Between Omaha and Texas Holdem*. Available at: https://www.888poker.com/magazine/strategy/omaha-poker/the-difference-between-omaha-and-texas-holdem.

Choosing a subset of flops to represent the whole game – PioSOLVER (2020). Available at: https://www.piosolver.com/blogs/news/62725637-choosing-a-subset-of-flops-to-represent-the-whole-game (Accessed: 22 February 2020).

CPRG (2016) *DeepStack*. Available at: https://www.deepstack.ai/ (Accessed: 5 September 2020).

CRPG (2020) *Annual Computer Poker Competition*. Available at: http://www.computerpokercompetition.org/ (Accessed: 1 September 2020).

Dinis Felix, L. P. R. (2008) 'An Experimental Approach to Online Opponent Modeling in Texas Hold'em Poker', *SBIA '08: Proceedings of the 19th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*. doi: 10.1007/978-3-540-88190-2_14.

Dr. Michael Bowling, Trevor Davis, Dustin Morrill, B. P. (2020) *CPRG*. Available at: https://poker.cs.ualberta.ca/ (Accessed: 1 September 2020).

Europe Poker Laws - Best Poker Sites in Europe Today (no date). Available at: https://www.pokerlaws.org/europe (Accessed: 22 February 2020).

Fischer, W. A. and Jost, J. W. (1989) 'Comparing structured and unstructured methodologies in firmware development', *Hewlett-Packard Journal*, 40(2), pp. 80–85.

Ganzfried, S. and Sandholm, T. (2015) 'Endgame solving in large imperfect-information games', *AAAI Workshop - Technical Report*, WS-15-07, pp. 20–29.

Glaister, D. (2007) 'Chips are down as man beats poker machine', *The Guardian*, 27 July. Available at: https://www.theguardian.com/technology/2007/jul/27/gambling?gusrc=rss&feed=technology.

Glatzer, J. (2015) 'Man vs. Machine: Pros Ahead of "Claudico" Over $450,000 After Seven Days', 4 May. Available at: https://www.pokernews.com/news/2015/05/man-vs-machine-pro-ahead-450k-21434.htm.

Glossary — Python 2.7.18 documentation (2020). Available at: https://docs.python.org/2/glossary.html#term-global-interpreter-lock (Accessed: 6 October 2020).

Grady, R. B. (1987) *Software Metrics: Establishing a Company-Wide Program*. Available at: https://www.amazon.com/Software-Metrics-Establishing-Company-Wide-1987-06-06/dp/B019NDPDCO (Accessed: 22 February 2020).

GTO+ – Making Game Theory Practical (no date). Available at: https://www.gtoplus.com/ (Accessed: 26 January 2020).

HM3 Official Page New Users (2020). Available at: https://www.holdemmanager.com/hm3/.

Hsu, J. (2017) 'Meet the New AI Challenging Human Poker Pros', *IEEE Spectrum*, 10

January. Available at: https://spectrum.ieee.org/automaton/artificial-intelligence/machine-learning/meet-the-new-ai-challenging-human-poker-pros.

Introduction | Socket.IO (2020). Available at: https://socket.io/docs/ (Accessed: 4 October 2020).

J. Maynard Smith, G. R. P. (1973) 'The Logic of Animal Conflict', *Nature*. doi: 10.1038/246015a0.

Jackson, Leyton-Brown, S. (2013) 'Imperfect Information Extensive Form: Definition, Strategies'. Stanford University, University of British Columbia. Available at: https://www.youtube.com/watch?v=dLWtcWPi84s.

jeskola.net/jesolver_beta/ (no date). Available at: http://jeskola.net/jesolver_beta/ (Accessed: 31 January 2020).

Johanson, M. (2015) *What is an intuitive explanation of counterfactual regret minimization? - Quora*. Available at: https://www.quora.com/What-is-an-intuitive-explanation-of-counterfactual-regret-minimization (Accessed: 22 February 2020).

John von Neumann, O. M. (1944) *Theory of Games and Economic Behavior*. 60th anniv. Princeton University Press. Available at: https://press.princeton.edu/books/paperback/9780691130613/theory-of-games-and-economic-behavior.

Koen, P. A. *et al.* (2002) 'Fuzzy Front End: Effective Methods, Tools, and Techniques', *The PDMA ToolBook for New Product Development*.

Line - Poker Terms Glossary | PokerStrategy.com (2020). Available at: https://www.pokerstrategy.com/glossary/Line/ (Accessed: 1 February 2020).

Malara, M. (2015) 'Brains vs. AI: Computer faces poker pros in no-limit Texas Hold'em', *UPI*, 25 April. Available at: https://www.upi.com/Science_News/2015/04/25/Brains-vs-AI-Computer-faces-poker-pros-in-no-limit-Texas-Holdem/6431429965877/.

McGraw-Hill, I. (2003) 'Quality Function Deployment', I.

MIT (2021) *MIT Pokerbots 2021*. Available at: https://pokerbots.org/ (Accessed: 1 September 2020).

MonkerWare.com - Abstraction (2020). Available at: https://monkerware.com/compare.html (Accessed: 26 January 2020).

MonkerWare.com - MonkerSolver (2020). Available at: https://monkerware.com/solver.html (Accessed: 26 January 2020).

Moore, G. E. (2004) 'Cramming more components onto integrated circuits', *Journal of Integrated Design and Process Science*, 8(3), pp. 49–60. Available at: https://drive.google.com/file/d/0By83v5TWkGjvQkpBcXJKT1I1TTA/view.

Moravčík, M. *et al.* (2017) 'DeepStack: Expert-level artificial intelligence in heads-up no-limit poker', *Science*, 356(6337), pp. 508–513. Available at:

https://science.sciencemag.org/content/356/6337/508.

Muthoo, A., Osborne, M. J. and Rubinstein, A. (1996) *A Course in Game Theory.*, *Economica*. doi: 10.2307/2554642.

Myers, G. J. (2005) *The Art of Software Testing, Second edition. Glenford J. Myers. Revised and updated by Tom Badgett and Todd M. Thomas, with Corey Sandler. John Wiley and Sons, New Jersey, U.S.A., 2004. ISBN: 0-471-46912-2, pp 234*, *Software Testing, Verification and Reliability*. doi: 10.1002/stvr.322.

Myerson, R. B. (1991) 'Game Theory: Analysis of Conflict'. Harvard University Press. doi: 10.2307/j.ctvjsf522.

Nash, J. F. (1949) 'Equilibrium points in n-person games'.

Nicola, S. (2019) 'Análise de valor'.

partypoker closes a further 121 bot accounts in July 2019; $76,267 and €88,351 seized (2019). Available at: https://www.partypoker.com/blog/en/partypoker-closes-a-further-121-bot-accounts-in-july-2019-76267-and-e88351-seized.html (Accessed: 19 February 2020).

Patrick W. Jordan, B. Thomas, Ian Lyall McClelland, B. W. (1996) *SUS—A Quick and Dirty Usability Scale*, *Usability Evaluation in Industry*. London: Taylor and Francis. doi: 10.5948/upo9781614440260.011.

Patvs (2013) *Statistic values explanation - Hold'em Manager Forums*. Available at: https://forums.holdemmanager.com/showthread.php?t=407991&p=1839781&viewfull=1#post1839781.

Piosolver.com - PioSOLVER (2020). Available at: https://www.piosolver.com/ (Accessed: 26 January 2020).

Poker, S. (no date a) *Simple 3-way*. Available at: https://simplepoker.com/en/Solutions/Simple_3-way (Accessed: 26 January 2020).

Poker, S. (no date b) *Simple Poker - Best GTO Solvers and Learning GTO Tools for Texas Holdem and Omaha Poker*. Available at: https://simplepoker.com/en/.

Poker Variance Calculator - Primedope (2020). Available at: https://www.primedope.com/poker-variance-calculator/ (Accessed: 16 February 2020).

PokerNews (2017) 'AI Beats Poker Pros in "Brains vs. AI" Event', January. Available at: https://www.pokernews.com/news/2017/01/poker-ai-beats-the-pros-26990.htm.

Pokerstars | Time bank in tournaments and cash games (2020). Available at: https://www.pokerstars.com/help/articles/trn-time-bank/63324/ (Accessed: 16 February 2020).

PokerTracker (2020). Available at: https://www.pokertracker.com/.

Polak, B. (2007) *Game Theory: Lecture 1*. 1. Available at:

https://oyc.yale.edu/economics/econ-159.

Poole, D. L., Mackworth, A. and Goebel, R. G. (1998) ‘Computational Intelligence and Knowledge’, *Computational Intelligence: A Logical Approach*, (Ci), pp. 1–22. Available at: https://www.cs.ubc.ca/~poole/ci.html.

Popper, N. (2011) ‘Three largest online poker sites indicted and shut down by FBI’, *Los Angeles Times*, (April 15). Available at: https://latimesblogs.latimes.com/money_co/2011/04/three-largest-online-poker-sites-indicted-and-shut-down-by-fbi.html.

Pork, A. (2004) *Agnostics vs. Zealots | PokerStove: Poker Software and Analysis*. Available at: http://web.archive.org/web/20120206021725/http://pokerstove.com/analysis/zealots.php (Accessed: 5 August 2020).

Rich, N. and Holweg, M. (2000) ‘Value Analysis and Value Engineering’. United Kingdom: Lean Enterprise Research Centre Cardiff, p. 32. Available at: https://www.urenio.org/tools/en/value_analysis.pdf.

Saaty, T. L. (1984) ‘The Analytic Hierarchy Process: Decision Making in Complex Environments’, in *Quantitative Assessment in Arms Control*. Springer US, pp. 285–308. doi: 10.1007/978-1-4613-2805-6_12.

Sandholm, T. (2020) *Tuomas W. Sandholm*. Available at: http://www.cs.cmu.edu/~sandholm/ (Accessed: 20 September 2020).

Sfetcu, N. (2014) *Game Preview*. Available at: https://books.google.pt/books?id=J1aAAwAAQBAJ.

Shor, M. (2006) *Cooperative Game - Game Theory .Net*. Available at: http://www.gametheory.net/dictionary/CooperativeGame.html.

SimplePostFlop (no date). Available at: https://simplepostflop.com/en/ (Accessed: 26 January 2020).

Skalinsky, B. D. and Malmuth, M. (1994) ‘Hold ’ em Poker For Advanced Players’.

Spice, B. and Allen, G. (2017) ‘Upping the Ante: Top Poker Pros Face Off vs. Artificial Intelligence’, *Carnegie Mellon University News*, 4 January. Available at: https://www.cmu.edu/news/stories/archives/2017/january/poker-pros-vs-AI.html.

Swains, H. (2006) ‘Moneymaker method can show the way to a fortune’, *The Times*, (Saturday, March 18). Available at: https://www.thetimes.co.uk/article/moneymaker-method-can-show-the-way-to-a-fortune-j7mpcn9jppq.

System Usability Scale (SUS) | Usability.gov (no date). Available at: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (Accessed: 22 February 2020).

System Usability Scale online with analytics | usabiliTEST (2020). Available at: https://www.usabilitest.com/system-usability-scale (Accessed: 22 February 2020).

Tammelin, O. *et al.* (2015) ‘Solving Heads-Up Limit Texas Hold ’ em’, (Ijcai), pp. 645–652.

Technical Details - PioSOLVER (2015). doi: 10.1201/9781420010138.axb.

Teófilo, F. (2016) ‘Methodologies and Tools for Creating Competitive Poker Playing Agents’.

Teófilo, L. F. (2011) ‘Estimating the Probability of Winning for Texas Hold’em Poker Agents’, *Proceedings 6th Doctoral Symposium on Informatics Engineering*, pp. 129–140.

The preflop solver - PioSOLVER (2015). Available at: https://www.piosolver.com/blogs/news/70944645-the-preflop-solver (Accessed: 2 January 2020).

Tipton, W. (2012) *Expert Heads Up No Limit Hold’em: Optimal And Exploitative Strategies (Poker Series) Volume 1: Optimal and Exploitative Strategies*. First edit. D & B Publishing.

Ulam, N. M. S. (1949) ‘The Monte Carlo Method’. Journal of the American Statistical Association, pp. 335–341.

Wedyan, F. and Abufakher, S. (2020) ‘Impact of design patterns on software quality: A systematic literature review’, *IET Software*. Institution of Engineering and Technology, 14(1), pp. 1–17. doi: 10.1049/iet-sen.2018.5446.

Woodall, T. (2003) *Conceptualising ‘Value for the Customer’: An Attributional, Structural and Dispositional Analysis*. Available at: https://www.researchgate.net/publication/228576532_Conceptualising_’Value_for_the_Customer’_An_Attributional_Structural_and_Dispositional_Analysis (Accessed: 22 February 2020).

Zinkevich, M. *et al.* (2009) ‘Regret minimization in games with incomplete information’, *Advances in Neural Information Processing Systems 20 - Proceedings of the 2007 Conference*, pp. 1–8.