UCLouvain

epl

École polytechnique de Louvain

Tricolor:

Computational Analysis and AI Exploration of an Early Hexagonal Board Game

Author: **Andrei Bourceanu**
Supervisors: **Eric Piette, Lisa Rougetet**
Readers: **Hélène Verhaeghe, Benoît Ronval**
Academic year 2025–2026
Master [120] in Computer Science

Contents

Summary

Tricolor is a combinatorial board game invented in 1930 by the Belgian mathematician Maurice Kraitchik. Notable for its early use of a hexagonal tiling, its stack-based piece mechanics, and its capture rules influenced by tile colors, the game constitutes an original object of study at the intersection of the history of mathematics, artificial intelligence, and computational game analysis. Despite its historical interest, Tricolor has not previously been the subject of a systematic computational study.

This thesis provides a first computational analysis of Tricolor. The rules of the game are formalized from historical sources and adapted into a precise model suitable for automated simulation and AI-based experimentation. An efficient game engine is implemented in C++, using compact representations of states and actions in order to support fast move generation, efficient state copying, and large-scale simulations.

Using this framework, the thesis studies several quantitative properties of the game. Simulations between Random Agents are used to estimate average game duration, branching factor, game tree complexity, state space complexity, player balance, and several dynamic gameplay metrics such as decisiveness, stability, and drama. The results show that Tricolor has a large search space, relatively long games, and unstable tactical dynamics, making it a non-trivial game from an artificial intelligence perspective. The thesis then identifies several strategic principles specific to Tricolor, including material control, positional power, stack concentration, and forced tactical sequences. These observations are used to design a heuristic evaluation function. Two classical game-playing agents are implemented and compared: an Alpha-Beta agent guided by this heuristic and a Monte Carlo Tree Search agent based on random simulations. Experimental results show that both agents strongly outperform random play, but that the Alpha-Beta agent clearly dominates the basic MCTS agent under the tested conditions. This difference is explained by the high branching factor, long game duration, and delayed tactical consequences created by stacking and forced moves.

Beyond agent performance, this work provides a computational tool for the future study of Tricolor. The developed framework can support artificial intelligence researchers, historians of games, mathematicians, and cultural heritage specialists in exploring the strategic properties of the game, testing historical interpretations, generating representative games, and comparing different styles of play. More broadly, this thesis illustrates how computational methods and artificial intelligence can contribute not only to playing historical games, but also to understanding, preserving, and studying them as mathematical and cultural objects.

Acknowledgments

I would like to thank Eric Piette (Université catholique de Louvain) for his valuable guidance and continuous support throughout this thesis. His suggestions and insights greatly contributed to the direction of this work, particularly regarding the game analysis methodology and the development of the AI agents. I am especially thankful for the relevant research papers and resources he provided, which helped me better understand previous work related to this subject and guided the development of this thesis. I also greatly appreciate the time he dedicated to reviewing this manuscript, as well as his assistance in improving the structure, clarity, and academic formulation of the paper.

I would like to thank Lisa Rougetet (Université de Bretagne Occidentale) for her collaboration on this work. She kindly provided historical materials related to Tricolor, including photographs of the original patent, as well as additional resources from historical books. These contributions were essential to the historical component of this study.

I would also like to thank Hélène Verhaeghe and Benoît Ronval (Université catholique de Louvain) for accepting to be part of my thesis evaluation, for taking the time to read my work, and for assessing it during the defense.

1

Introduction

1.1 Context and Problem Statement

Board games have long been studied not only as recreational artifacts, but also as objects of mathematical, historical, and computational interest [1] [2]. In recent years, advances in artificial intelligence and general game playing have enabled the large-scale computational analysis of traditional and historical games [3]. Research initiatives such as the GameTable COST Action investigate how AI-driven methods can contribute to the preservation, reconstruction, and understanding of tabletop game heritage [4]. In particular, recent work on the Roman board game Ludus Coriovalli demonstrated how AI-based simulations can help identify plausible rule systems and strategic properties for historically documented games [5].

Tricolor is a combinatorial board game invented in 1930 by the Belgian mathematician Maurice Kraitchik [6]. The game is characterized by a hexagonal board, stack-based mechanics, and movement and capture rules influenced by the color of the board tiles. Tricolor occupies a particular place in the history of board games, as it is considered one of the earliest known games to use a hexagonal tiling [7]. Despite its historical originality and unusual mechanics, the game has received very limited modern scientific attention. Existing sources mainly consist of the original patent and a small number of historical descriptions and example games [7] [8]. From a computational perspective, Tricolor presents several characteristics commonly associated with difficult combinatorial games. The game combines a potentially large branching factor, long game durations, forced tactical interactions, and stack-based piece mechanics that significantly increase the complexity of state representation and strategic planning. These properties make Tricolor an interesting case study for classical game-playing algorithms such as Alpha-Beta pruning and Monte Carlo Tree Search [9] [10].

Beyond the sole objective of developing competitive AI agents, this work also investigates how computational methods can contribute to the study of historical mathematical games. By formalizing the rules of Tricolor, implementing an efficient simulation framework, and performing large-scale experiments, this thesis aims to provide a foundation for future studies in artificial intelligence, combinatorial game theory, digital humanities, and computational game heritage research [3] [4] [11].

This thesis therefore studies Tricolor from both a computational and strategic perspective, with the objective of characterizing its complexity, identifying meaningful heuristics, and evaluating the effectiveness of different AI approaches for playing and analyzing the game.

1.2 Motivation

The effectiveness of artificial intelligence techniques for board games strongly depends on the structural properties of the game being studied. While some games can be explored efficiently through deep search, others generate search spaces so large that exhaustive analysis rapidly becomes impractical [9] [10]. Understanding the computational characteristics of a game is therefore an important step when designing efficient AI agents.

Tricolor presents several features that make it particularly challenging from a computational perspective. Unlike classical board games where each piece occupies a single independent position, Tricolor allows pieces from both players to be stacked together on the same tile. Consequently, the state of the game depends not only on the location of pieces, but also on stack composition, ownership, and stack size. This considerably increases the complexity of state representation and move evaluation.

Another important aspect of the game is the presence of forced tactical interactions. Whenever a capture or imprisonment move is available, the player is required to perform such a move. This constraint strongly shapes gameplay and strategic planning, since players may be forced to weaken otherwise stable defensive structures. As a result, local tactical situations can produce long-term strategic consequences that are difficult to evaluate. The geometry and coloring of the board also contribute significantly to the complexity of the game. Since the color of a tile directly modifies the attack and defense power of stacks, positional control becomes a central component of gameplay. Players must therefore balance several competing objectives simultaneously, including mobility, stack growth, tactical threats, and territorial control.

These mechanics collectively generate a large search space characterized by high branching factors and long game durations. Under such conditions, exhaustive search rapidly becomes infeasible, making Tricolor an interesting benchmark for evaluating the strengths and limitations of classical game-playing algorithms such as Alpha-Beta pruning and Monte Carlo Tree Search [9] [10].

Beyond the development of competitive AI agents, another motivation of this work is to provide a computational framework that can support the systematic study of Tricolor. By formalizing the rules of the game and implementing efficient simulation and analysis tools, this thesis aims to facilitate large-scale experimentation that would be impractical to perform manually. Such a framework can support future investigations of the game from strategic, mathematical, historical, and artificial intelligence perspectives.

1.3 Research Questions

This thesis investigates several research questions related to the computational analysis and AI-based study of Tricolor.

The first question concerns the formalization and implementation of the game:

- How can Tricolor be modeled computationally in a way that enables efficient simulation and large-scale experimentation?

A second question concerns the structural complexity of the game:

- What are the main computational characteristics of Tricolor, such as its average game duration, branching factor, game tree complexity, and state space complexity?

Another objective of this work is to investigate the strategic properties emerging from the rules of the game:

- What types of strategic patterns and heuristics can be identified through simulation and analysis?

This thesis also studies the effectiveness of classical game-playing algorithms in the context of Tricolor:

- How do Alpha-Beta pruning and Monte Carlo Tree Search perform on a game characterized by stack-based interactions, forced tactical moves, and large search spaces?

Finally, this work explores the broader role of computational methods in the study of historical games:

- To what extent can simulation frameworks and AI-driven analysis support future mathematical, strategic, and historical investigations of Tricolor?

1.4 Contributions

This thesis makes several contributions to the computational study of Tricolor.

First, the rules of the game are formally analyzed and adapted into a computational framework suitable for simulation and AI-based experimentation. In particular, an additional draw condition is introduced in order to guarantee game termination during large-scale simulations.

Second, an efficient implementation of the game is developed in C++. The proposed framework includes compact representations for board states and actions, efficient move generation mechanisms, and a graphical interface allowing games to be visualized and played by human users or AI agents.

Third, this work provides an empirical study of several computational and gameplay characteristics of Tricolor. Using large-scale simulations, multiple metrics are analyzed, including game duration, branching factor, game tree complexity, state space complexity, decisiveness, stability, and drama.

Fourth, this thesis investigates strategic properties emerging from the mechanics of the game. In particular, several heuristics related to stack management, positional control, and tactical interactions are identified and analyzed.

Fifth, two classical game-playing approaches are implemented and evaluated on Tricolor: Alpha-Beta pruning and Monte Carlo Tree Search. Their performance is experimentally compared through large sets of simulated games.

Finally, beyond the sole objective of designing AI agents, this work introduces a computational framework intended to support future investigations of Tricolor from mathematical, strategic, and historical perspectives. The implemented tools and analyses aim to facilitate interdisciplinary research on this historically significant game.

1.5 Methodology Overview

This thesis follows a methodology combining formal modeling, software implementation, experimental analysis, and AI-based evaluation.

The work begins with a formal analysis of the rules of Tricolor in order to obtain a computational representation of the game suitable for simulations and automated experimentation. Particular attention is given to the representation of stack-based interactions, move legality, and game termination conditions.

Based on this formalization, a dedicated game engine is developed in C++. The implementation is designed to support efficient move generation, rapid state manipulation, and large-scale simulations required for experimental analysis and AI search algorithms.

The game is then studied empirically through automated self-play simulations. Large numbers of generated games are analyzed in order to estimate structural and gameplay-related characteristics of Tricolor. These experiments provide quantitative information about the complexity and dynamics of the game.

In parallel, gameplay observations and experimental analyses are used to identify recurring strategic patterns and important positional features. These observations are subsequently incorporated into heuristic evaluation mechanisms for AI agents.

Finally, the thesis evaluates different classical game-playing approaches on Tricolor. Experimental comparisons between agents are performed in order to analyze the behavior, strengths, and limitations of the considered algorithms in the context of the game.

1.6 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 presents the historical background of Tricolor and discusses related work on historical games, combinatorial game analysis, and artificial intelligence techniques applied to board games.

Chapter 3 introduces the formal rules of Tricolor and describes the computational representation of the game used throughout this work.

Chapter 4 presents the implementation of the game engine, including the representation of board states and actions, move generation mechanisms, and the graphical interface developed for simulations and gameplay visualization.

Chapter 5 studies the computational and gameplay characteristics of Tricolor through large-scale simulations. Several metrics related to complexity, game dynamics, and gameplay properties are analyzed experimentally.

Chapter 6 investigates strategic aspects of the game and introduces heuristic observations derived from gameplay analysis and experimentation.

Chapter 7 presents the artificial intelligence approaches implemented in this work, including Alpha-Beta pruning and Monte Carlo Tree Search.

Chapter 8 evaluates the performance of the implemented agents through experimental comparisons and discusses their respective strengths and limitations in the context of Tricolor.

Finally, Chapter 9 concludes the thesis and discusses future perspectives for the computational, mathematical, and historical study of Tricolor.

2

Historical Background and Related Work

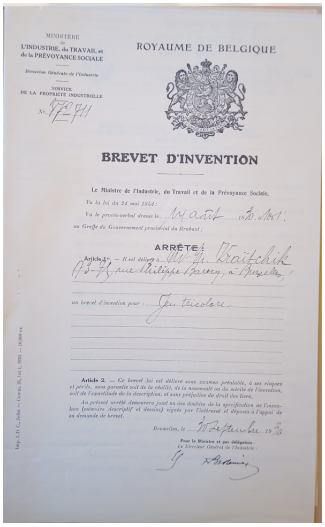
2.1 Maurice Kraitchik and the Origins of Tricolor

Tricolor was invented in 1930 by the Belgian mathematician Maurice Kraitchik [6]. Kraitchik was an important figure in recreational mathematics during the first half of the twentieth century and is notably known for his work on mathematical puzzles, number theory popularization, and mathematical recreations.

The game was introduced through a Belgian patent entitled Tricolor [7]. The original patent document, reproduced in 2.1, presents the rules of the game, the board geometry, and the initial arrangement of the pieces. Tricolor was later included in Kraitchik's book *La mathématique des jeux ou récréations mathématiques* [8], which is discussed in more detail in the next section.

One of the distinctive aspects of Tricolor is its use of a hexagonal board whose colored tiles directly influence gameplay. A representation of the game board and pieces is shown in 2.2. The game also introduces stack-based mechanics and forced tactical interactions, making it structurally different from many traditional abstract strategy games of the same period.

Historically, Tricolor occupies a particular place among board games because it is considered one of the earliest known games to use a hexagonal tiling [7]. While hexagonal boards would later become more common in abstract strategy games, they remained relatively uncommon at the time of the game's creation. Tricolor therefore represents an early example of experimentation with alternative board geometries and movement systems.



Kraïtchik, Maurice (1930) Tricolor. Brevet n°372711, Royaume de Belgique

Figure 2.1: Photo of the official patent certificate for the invention of Tricolor

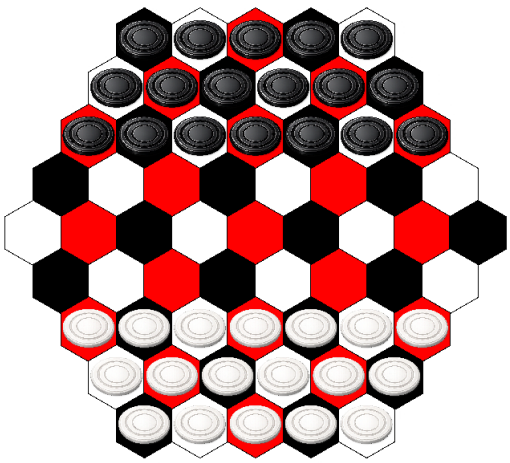


Figure 2.2: Tricolor board and pieces

2.2 Historical Description of the Game

Kraitchik's later presentation of Tricolor in *La mathématique des jeux ou récréations mathématiques* provides the main historical description of the game beyond the patent [8]. In this book, Tricolor is included among a broader collection of mathematical recreations, which situates it within Kraitchik's general interest in games, puzzles, and recreational mathematics.

This source is particularly useful because it does more than state the rules. Kraitchik introduces a notation for recording moves and gives examples of complete or partial games. These examples show how positions and sequences of moves were historically represented, and they provide concrete material for interpreting the dynamics of play. One such example is reproduced in 2.3.

N°	Blancs	Noirs	N°	Blancs	Noirs	N°	Blancs	Noirs
1	6 — 12	51 — 44	12	17 — 18	56 — 50	23	$31^2 \times 48$	50 × 48
2	13 — 12	45 — 44	13	12 — 38	44 — 21	24	31 — 33	$48^2 \times 33$
3	1 — 7	57 — 52	14	38^1 — 30	21^1 : 30	25	18^2 : 33	$48^2 \times 33$
4	2 — 7	58 — 52	15	15^2 : 30	47^2 : 30	26	18 — 11	33 — 24
5	8 — 3	60 — 59	16	3 × 21	59 × 38	27	11 — 10	24^1 — 17
6	9 — 3	53 — 59	17	21 : 30	38 : 30	28	10 : 17	24 : 17
7	4 — 10	46 — 47	18	15 — 9	30^3 : 9		—	gagne
8	5 — 10	54 — 47	19	7 : 9	30 — 39			
9	14 — 15	61 — 55	20	9 — 10	39 — 52			
10	16 — 15	48 — 55	21	10 — 31	55^1 — 48			
11	11 — 18	49 — 50	22	31^2 : 48	$55^2 \times 48$			

Figure 2.3: Example of a Tricolor game played between two players

The book also discusses selected endgame situations, especially cases in which one player has a material advantage. These analyses are not systematic, but they reveal that the game was already considered by its inventor as an object of strategic reasoning rather than merely as a set of rules.

In the present thesis, these historical descriptions are used as reference material for the formalization of the game. They help clarify how the rules should be interpreted and provide examples that can be compared with the computational model introduced in the following chapters.

2.3 Hexagonal Board Games and Related Games

Tricolor is closely related to the broader history of abstract strategy games that experiment with non-square board geometries. Most traditional European board games, such as Chess, Draughts, or Checkers, are played on square grids. By contrast, Tricolor uses a hexagonal tiling, which gives each cell six neighboring directions and creates movement patterns that differ from those found on square boards.

This use of hexagonal geometry is one of the reasons why Tricolor is historically notable. Later abstract strategy games would also explore hexagonal boards, most famously Hex, invented independently by Piet Hein and John Nash in the twentieth century [12].

However, Tricolor predates many better-known hexagonal board games and therefore represents an early example of this design choice.

From a mechanical point of view, Tricolor also shares similarities with games involving stacks of pieces. The most direct historical comparison is Laska, also known as Lasca, invented by Emanuel Lasker in 1911 [13]. Like Tricolor, Laska allows pieces to be stacked and captured through movement-based interactions. Kraitchik explicitly presents Tricolor as being inspired by Laska [7].

However, Tricolor differs from Laska in several important ways. Its hexagonal board changes the geometry of movement, while its colored tiles introduce a positional power system that affects both attack and defense. In addition, the combination of stacking, imprisonment, capture, and forced tactical moves gives Tricolor a distinct strategic identity.

For these reasons, Tricolor can be viewed as part of a broader tradition of abstract strategy games that explore alternative board structures and piece interactions, while also presenting original features that justify a dedicated computational study.

2.4 Artificial Intelligence in Combinatorial Games

Combinatorial board games have played an important role in the development of artificial intelligence. Because such games usually have explicit rules, discrete states, and well-defined outcomes, they provide suitable environments for studying search, decision-making, and strategic reasoning [9].

Classical game-playing approaches often rely on tree search. Minimax and Alpha-Beta pruning are among the most established techniques for deterministic two-player games with perfect information [9]. These methods evaluate possible sequences of moves and try to identify actions that maximize the expected outcome for one player while minimizing the opponent's chances. Their effectiveness depends heavily on the branching factor of the game, the depth that can be reached within a limited computation time, and the quality of the evaluation function used at non-terminal positions.

Monte Carlo Tree Search is another influential approach, especially for games where exhaustive search is difficult [10]. Instead of exploring the game tree uniformly, MCTS uses simulations to estimate the value of actions and gradually focuses the search on promising parts of the tree. This approach has been successful in several complex games, particularly when combined with strong evaluation mechanisms or learning-based components.

More recent approaches combine search with machine learning, and especially deep learning. Systems such as AlphaGo and AlphaZero demonstrated that neural networks trained through self-play can be used to guide search and evaluate positions in complex board games [14] [15]. In such systems, a policy network helps select promising actions, while a value network estimates the expected outcome of a position. This combination can lead to very strong agents, even when handcrafted heuristics are limited.

In principle, similar learning-based approaches could be applied to Tricolor. A neural agent could be trained through self-play in order to learn positional evaluations, strategic patterns, and move-selection policies directly from generated games. This direction is particularly appealing because Tricolor contains complex stack-based interactions and long-term tactical dependencies that may be difficult to capture fully through manually designed heuristics.

However, deep learning approaches also require substantial computational resources, large numbers of self-play games, careful neural-network design, and extensive hyperparameter tuning. In the case of Tricolor, the stack-based representation of the board also raises additional modeling questions, since the state of a tile is not limited to the presence of a single piece but may include several pieces of both players. This makes the design of an appropriate neural input representation less straightforward than for many classical board games.

For these reasons, this thesis focuses primarily on classical search-based methods, namely Alpha-Beta pruning and Monte Carlo Tree Search. These approaches are well suited for a first computational study of Tricolor because they require fewer training resources, are easier to interpret, and make it possible to analyze the impact of handcrafted heuristics on agent performance. Learning-based methods remain a promising direction for future work, once a reliable implementation, baseline agents, and empirical understanding of the game have been established.

The efficiency of all these algorithms depends strongly on the structure of the game being studied. Branching factor, average game length, state representation, move-generation cost, and the quality of position evaluation all influence the practical performance of AI agents. These considerations are directly relevant for Tricolor, whose stack-based mechanics and large action spaces create specific challenges for both classical search methods and future learning-based approaches.

2.5 Computational Analysis of Historical Games

Recent research has shown that computational methods can be used not only to design strong game-playing agents, but also to study historical and traditional games as cultural, mathematical, and strategic objects. In this context, artificial intelligence provides tools for formalization, simulation, comparison, reconstruction, and preservation.

A central example of this direction is the Ludii general game system, which provides a framework for describing, playing, and analyzing a wide range of games using a formal ludeme-based representation [16]. This framework is closely connected to the Ludii Games Database, an open-access resource designed to document traditional board games, their rulesets, ludemes, and historical evidence within a unified scholarly structure [17]. Together, these tools support both computational analysis and cultural research on games.

Several recent studies illustrate how such approaches can be applied to historical games. A Ludii-based analysis of the French Military Game showed how computational tools can validate and extend historical claims about a nineteenth-century strategic game [18]. Work on Ludus Latrunculorum used computational methods to evaluate possible reconstructions of an ancient Roman game from incomplete textual and archaeological evidence [19]. Similarly, research on Mesopotamian games applied AI-simulated play to compare historical board geometries and measure gameplay properties such as duration, player advantage, and drama [20].

More recently, the study of Ludus Coriovalli combined archaeological use-wear analysis with AI-driven simulations to identify plausible rules for a Roman-era board game [21]. This work demonstrates how computational game analysis can contribute directly to archaeological interpretation by comparing simulated gameplay patterns with physical

evidence. Such studies show that AI can serve not only as a means of producing strong players, but also as an investigative tool for understanding historical play.

This research direction is also supported by the GameTable COST Action, which aims to connect artificial intelligence, archaeology, history, mathematics, and game studies around the computational study of tabletop game heritage [22]. Reports from GameTable meetings emphasize the need for shared frameworks, case studies, and communication tools that can bridge AI research and cultural heritage practices [23].

The present thesis follows this broader line of work, but with a different emphasis. In the case of Tricolor, the main rules are known from historical sources, so the goal is not primarily to reconstruct an unknown game. Instead, the objective is to formalize, implement, simulate, and analyze the game in order to better understand its computational and strategic properties. The resulting framework can then support future studies of Tricolor from mathematical, historical, and artificial intelligence perspectives.

2.6 Summary

This chapter situated Tricolor within its historical and scientific context. It first introduced Maurice Kraitchik and the origins of the game, then described the main historical sources in which Tricolor appears, including the Belgian patent and Kraitchik's later presentation in *La mathématique des jeux ou récréations mathématiques*.

The chapter also positioned Tricolor in relation to other abstract strategy games, especially games using non-square geometries or stack-based mechanics such as Laska. This comparison highlights both the historical originality of Tricolor and the specific features that distinguish it from related games.

Finally, the chapter reviewed the main artificial intelligence approaches relevant to this thesis, from classical search methods such as Alpha-Beta pruning and Monte Carlo Tree Search to more recent learning-based approaches. It also presented recent work on the computational analysis of historical games, showing how AI-driven methods can support the study, reconstruction, and preservation of game heritage.

The next chapter presents the rules of Tricolor in a precise form suitable for implementation, simulation, and automated analysis.

3

Formal Rules of Tricolor

The purpose of this chapter is to present the rules of Tricolor in a precise form suitable for computational modeling. While the historical sources describe the game in its original context, the present chapter focuses on the two-player version considered throughout this thesis. The objective is to define the board, pieces, legal moves, capture mechanisms, and game-ending conditions in a way that can be directly used for simulation and AI-based analysis.

3.1 Board, Pieces and Initial Setup

Tricolor is played on a hexagonal board composed of 61 tiles. Each tile has one of three colors: white, black, or red. These colors are not merely decorative: they directly influence the strength of pieces placed on them, as explained in Section 3.3.

In the two-player version considered in this thesis, each player starts with 18 pieces. Player 1 controls the white pieces and Player 2 controls the black pieces. The initial arrangement of the pieces is shown in 3.1. By convention, Player 1 always makes the first move.

A central feature of Tricolor is that several pieces may occupy the same tile. Such a group of pieces is called a *stack*. A stack may contain pieces of one or both players. The player who controls a stack is determined by the color of the piece on top of it. If the top piece is white, the stack is controlled by Player 1; if the top piece is black, the stack is controlled by Player 2. Pieces located below an opponent's top piece are considered *imprisoned*: they remain on the board, but they do not control the stack. An example of a stack is shown in 3.2.

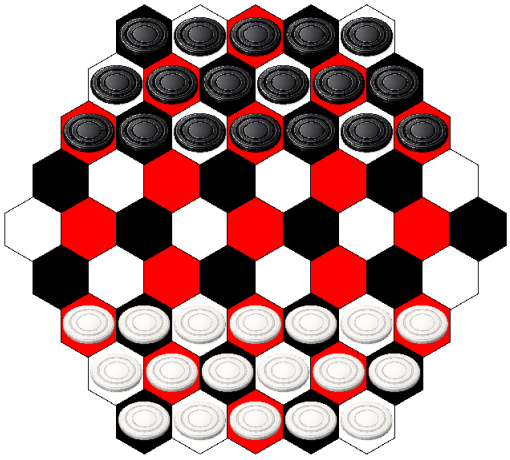


Figure 3.1: Initial board setup

Figure 3.2: Example of a stack controlled by the player with the black pieces

3.2 Legal Moves

Players alternate turns. On their turn, a player must choose one stack they control and move one or more pieces from the top of that stack to another tile.

A move is performed in a straight line along one of the six directions of the hexagonal grid, as illustrated in 3.3. The maximum distance that may be travelled depends on the number of friendly pieces selected in the moving group, up to a maximum distance of three tiles. More precisely, if the moving group contains one friendly piece, it may move at most one tile; if it contains two friendly pieces, it may move at most two tiles; and if it contains three or more friendly pieces, it may move at most three tiles. Opponent pieces contained in the moving group do not increase this movement distance.

All intermediate tiles between the origin and the destination must be empty. If at least one intermediate tile is occupied, the move is not legal.

The destination tile determines the effect of the move. There are three possible cases.

First, if the destination tile is empty, the selected pieces are simply moved to that tile.

Second, if the destination tile is controlled by the same player, the moved pieces are added to the existing stack. The moving player keeps control of the resulting stack, since their pieces are placed above any opponent pieces already present in the destination stack.

Third, if the destination tile is controlled by the opponent, the move is only legal if the moving stack is strong enough to defeat the defending stack. This interaction is governed

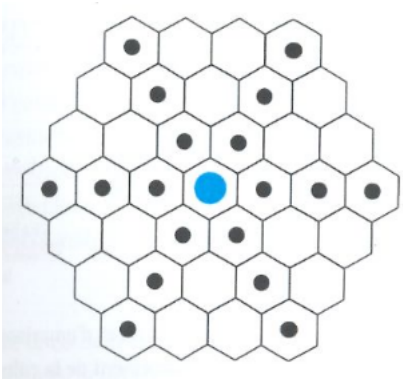


Figure 3.3: All six directions a stack can move in.

by the notion of *power*, described in the next section.

3.3 Power, Capture, and Imprisonment

The strength of a stack depends on two elements: the number of friendly pieces involved and the color of the tile on which the stack is located. Each tile color has an associated value:

- white tiles have value 1
- black tiles have value 2
- red tiles have value 3

The power of a stack is computed as:

$$\min(3, n) * c$$

where n is the number of friendly pieces in the stack and c is the value of the tile color. The function $\min(3, n)$ reflects the fact that no more than three pieces contribute to the power of a stack.

When a player moves onto a tile controlled by the opponent, the power of the attacking group is compared with the power of the defending stack. Three situations may occur.

If the attacking power is strictly greater than twice the defending power, the move is a *capture*. In this case, the opponent's pieces located on the destination tile are removed from the game. The attacking pieces then occupy the destination tile. An example of such a capture is shown in 3.4.

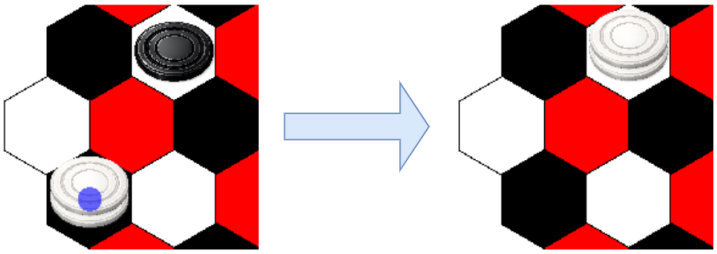


Figure 3.4: Example of a capture move made by the player with the white pieces

If the attacking power is strictly greater than the defending power, but not strictly greater than twice the defending power, the move is an *imprisonment*. In this case, the defending pieces are not removed from the board. Instead, the attacking pieces are placed on top of them, and the attacking player gains control of the stack. An example of imprisonment is shown in 3.5.

If the attacking power is less than or equal to the defending power, the move is illegal. The attacking player may not move onto that tile.

Tricolor also includes an important *forced-move* rule. If a player has at least one legal move that captures or imprisons an opponent's stack, then the player must choose one such move. If several capture or imprisonment moves are available, the player may choose freely among them. This rule has a strong influence on the strategy of the game, since players may be forced into tactical exchanges even when they would prefer to avoid them.

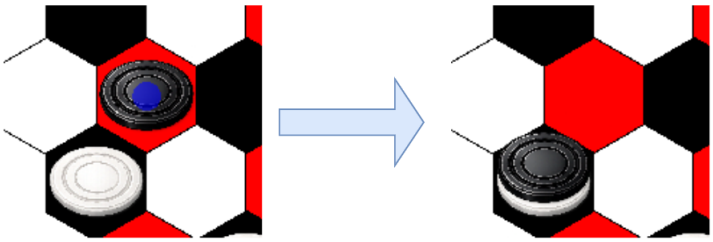


Figure 3.5: Example of an imprisonment move made by the player with the black pieces

3.4 End of the Game

A game of Tricolor ends when the player whose turn it is has no legal move available. In that case, the opposing player wins. This may happen because all of the player's pieces have been captured or imprisoned, or because their remaining controlled stacks have no legal movement available. An example of a terminal position is shown in 3.6.

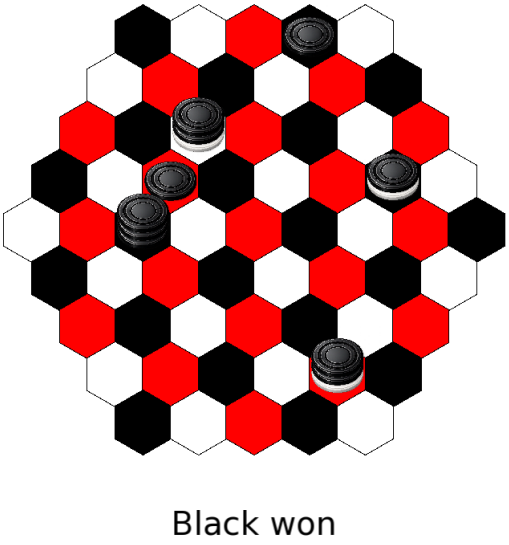


Figure 3.6: Example with the final position of a game won by the player with the black pieces

The original rules do not include a general mechanism preventing a game from continuing indefinitely. This is problematic for computational analysis, since automated simulations may produce extremely long games if both players avoid decisive interactions. For this reason, this thesis introduces an additional draw condition: if no capture has occurred during the last 100 turns, the game is declared a draw.

Here, a turn corresponds to a single move made by one player. Therefore, 100 turns correspond to 50 moves by each player. This additional rule is not intended to modify

the historical nature of the game, but to make large-scale simulations and AI experiments well-defined and computationally practical.

The rules presented in this chapter define the formal version of Tricolor used throughout the rest of this thesis. The next chapter explains how these rules are implemented efficiently in order to support simulation, move generation, and AI-based search.

4

System Design and Implementation

The previous chapter defined the formal rules of Tricolor used in this thesis. This chapter describes how these rules were implemented in order to support simulation, gameplay visualization, and AI-based analysis. The implementation was developed in C++, with the objective of obtaining an efficient and flexible framework for generating legal moves, applying actions, copying game states, and running large numbers of games.

4.1 Design Objectives and Architecture

The implementation of Tricolor was designed with three main objectives.

First, the game engine must accurately reproduce the rules presented in Chapter 3. This includes stack ownership, movement constraints, capture and imprisonment mechanisms, forced tactical moves, and game-ending conditions.

Second, the implementation must support efficient simulation. This is essential because the analysis presented in the following chapters relies on large numbers of automatically generated games. It is also important for AI agents such as Alpha-Beta and Monte Carlo Tree Search, which repeatedly generate actions, apply moves, and evaluate resulting states.

Third, the system must allow game states to be copied quickly. Search-based algorithms explore many possible continuations from a given position, and therefore require frequent duplication of board states. A compact representation is particularly useful for reducing both memory usage and computation time.

The implementation is organized around four main components. The board state stores the complete information required to describe a position. The action structure represents a legal move from one tile to another. The game engine is responsible for generating legal actions, applying moves, and detecting terminal states. Finally, agents use the game engine to select actions, either randomly, through human input, or through AI algorithms introduced later in the thesis.

4.2 Board State and Action Representation

The board of Tricolor contains 61 tiles, as shown in 4.1. A natural representation of a board state is therefore an array of 61 elements, where each element stores the information associated with one tile. Since a tile may contain several white and black pieces, and since the controlling player may depend on the top piece, each tile must encode three types of information: the number of white pieces, the number of black pieces, and the owner of the stack.

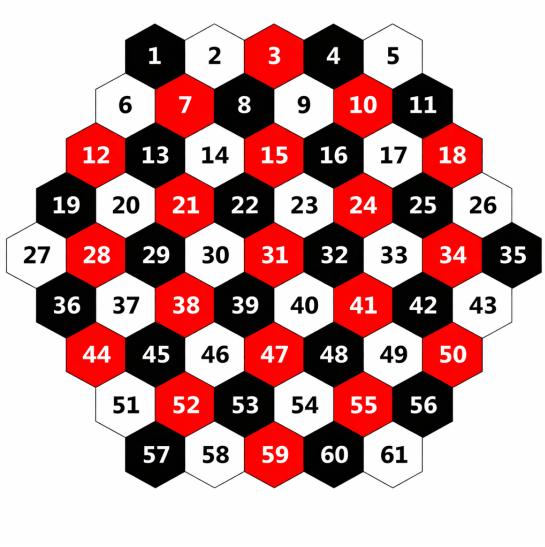


Figure 4.1: Board with numbered tiles from 1 to 61

Each player starts with 18 pieces, and the number of pieces can only decrease during the game. Therefore, the number of white pieces and the number of black pieces on a tile can each be represented using 5 bits, since values from 0 to 18 must be stored. The owner of a non-empty tile can be represented using 1 additional bit. Empty tiles do not require a separate owner value, since their neutrality can be inferred from the absence of pieces.

In the implemented representation, each tile is stored as a 16-bit integer. The number of white pieces, the number of black pieces, and the owner bit are encoded using bit masks. A visual representation of the board state can be seen in 4.2. This compact representation makes it possible to retrieve and modify tile information using bitwise operations, which are efficient and avoid more expensive object-based structures.

The player to move is stored separately as a Boolean value. Consequently, the size of a board state is:

$$61 \cdot 16 \text{ bits} + 8 \text{ bits} = 61 \cdot 2 \text{ bytes} + 1 \text{ byte} = 123 \text{ bytes}$$

This small memory footprint makes board states inexpensive to copy, which is especially important for search algorithms.

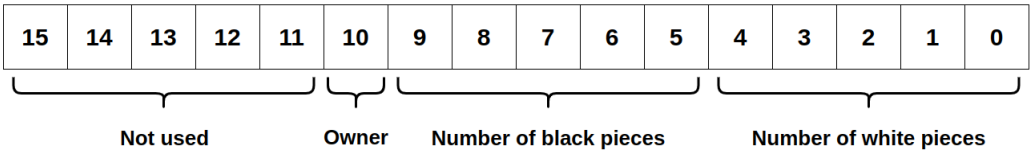


Figure 4.2: 16-bit integer visual representation masking the state of the board

Actions are represented in a similarly compact way. A move in Tricolor is fully described by three pieces of information: the origin tile, the destination tile, and the number of pieces moved. Since the board contains 61 tiles, both the origin and destination indices can be stored using one byte each. The number of moved pieces can also be stored in one byte. An action is therefore represented using only:

$$8 \text{ bits} + 8 \text{ bits} + 8 \text{ bits} = 1 \text{ byte} + 1 \text{ byte} + 1 \text{ byte} = 3 \text{ bytes}$$

This representation is sufficient to apply a move to a board state, while keeping the action structure small enough for efficient storage and manipulation during move generation and search.

4.3 Move Generation and Game Simulation

The move generator is responsible for producing all legal actions available from a given board state. To do this, it first identifies all tiles controlled by the player to move. For each controlled stack, it considers all possible numbers of pieces that may be moved from the top of the stack, all six directions of the hexagonal grid, and all legal distances allowed by the number of friendly pieces in the moving group.

For each candidate movement, the generator verifies the constraints defined in Chapter 3. The destination must be reachable in a straight line, and all intermediate tiles must be empty. If the destination tile is empty or controlled by the moving player, the move is legal. If the destination tile is controlled by the opponent, the attacking and defending powers are computed in order to determine whether the move is a capture, an imprisonment, or an illegal move.

The forced-move rule is also handled during move generation. If at least one legal capture or imprisonment is available, then only capture and imprisonment moves are returned as legal actions. Otherwise, all ordinary legal moves are returned. This ensures that agents cannot select moves that violate the rules of Tricolor.

The game simulation loop uses this move generator at each turn. Starting from the initial board position, the current player selects one of the legal actions. The action is then applied to the board state, the current player is updated, and the game checks whether a terminal condition has been reached. A player loses if they have no legal move available on their turn. In addition, the implementation applies the draw condition introduced in Chapter 3: if no capture occurs during 100 consecutive turns, the game is declared a draw.

This simulation process is used both for complete games between agents and for internal searches performed by AI algorithms. Its efficiency is therefore central to the experimental results presented in the following chapters.

4.4 Agents and User Interface

Several basic agents were implemented to interact with the game engine.

The Random Agent selects uniformly among the legal actions available in the current position. Although simple, this agent is useful for testing the correctness of the implementation and for generating large numbers of games. It also provides a basic baseline against which stronger agents can later be compared.

The Human Agent allows a human player to select actions manually. This is useful for exploring the game, checking rule behavior in specific situations, and playing against either another human or an AI agent.

A graphical user interface was developed using the SFML library. The interface displays the board, the pieces, and the current position, and allows games to be visualized during play. This visual component is useful not only for human play, but also for debugging and for interpreting the behavior of agents during simulations.

The implementation therefore serves two complementary purposes. It provides an efficient experimental framework for automated simulations and AI search, while also offering a visual tool for exploring Tricolor positions and gameplay. The framework described in this chapter provides the basis for the quantitative analysis of Tricolor presented in the next chapter.

5

Complexity and Game Metrics Analysis

The previous chapter described the implementation used to simulate games of Tricolor efficiently. This chapter uses this framework to analyze the computational and gameplay properties of the game. The objective is not yet to evaluate strong AI agents, but rather to obtain a general quantitative understanding of Tricolor as a combinatorial board game.

The analysis presented here is based on large-scale simulations between Random Agents. This choice makes it possible to estimate general properties of the game without introducing strategic assumptions specific to a given AI algorithm. Although random play does not represent expert-level gameplay, it provides a useful first approximation of the game's depth, branching factor, balance, and dynamic behavior. These results also serve as a baseline for the agent evaluations presented later in the thesis.

Several of the metrics discussed in this chapter are inspired by previous work on game analysis, game design, and general board game concepts [1] [2] [9] [24]. In particular, gameplay metrics such as branching factor, game length, decisiveness, stability, and drama help characterize not only the computational difficulty of Tricolor, but also some of its structural and aesthetic properties as a game.

5.1 Experimental Protocol

In order to estimate the main properties of Tricolor, one million games were simulated between two Random Agents. In each position, a Random Agent selects uniformly among all legal moves returned by the move generator. If the forced-move rule applies, the agent therefore selects uniformly among the available capture or imprisonment moves only.

A *turn* is defined as a single move made by one player. Thus, a sequence of two turns corresponds to one move by each player. This convention is used consistently throughout the thesis, including when comparing Tricolor with other games.

For each simulated game, several quantities were recorded:

- the total number of turns before the game ended

- the number of legal actions available at each turn
- the final outcome of the game
- the evolution of position evaluations during play
- the number of repeated states
- the number of occupied board sites
- the average movement distance
- the number of pieces remaining on the board
- several game-dynamics metrics such as decisiveness, stability, and drama

Random simulations have two main advantages. First, they are computationally inexpensive, which makes it possible to generate a very large sample of games. Second, they do not depend on the design choices of a specific AI strategy. This makes them suitable for estimating broad structural properties of the game.

However, the results should be interpreted carefully. Random play may produce positions that strong players would avoid, and it may fail to reveal strategic patterns that only emerge under more rational play. For this reason, the metrics presented in this chapter should be understood as a baseline characterization of Tricolor, rather than as a definitive description of expert play.

5.2 Computational Complexity Metrics

5.2.1 Game Duration

The first metric measures the duration of a game in number of turns. This is important for AI agents because game length directly affects the size of the search tree and the cost of simulations. A game that tends to last many turns is generally harder to analyze exhaustively than a short game with the same branching factor.

The distribution of game durations obtained from the simulations is shown in 5.1. The average duration is approximately 177 turns, with a standard deviation of 69 turns.

This result shows that Tricolor games can be relatively long. For comparison, Shannon's classical estimate for Chess uses an average game length of about 40 moves per player, corresponding to about 80 individual player moves under the convention used here [1]. Under random play, Tricolor therefore appears to have a greater average depth than this classical estimate for Chess.

The high standard deviation also indicates that game duration varies substantially. Some games end quickly because captures and imprisonments rapidly reduce one player's mobility, while others continue for much longer due to extended movement, delayed confrontation, or repeated positional reorganization.

This variability is important for AI. In Monte Carlo Tree Search, long games make random rollouts more expensive. In Alpha-Beta search, long games make it less likely that the search reaches terminal states, increasing the importance of heuristic evaluation functions.

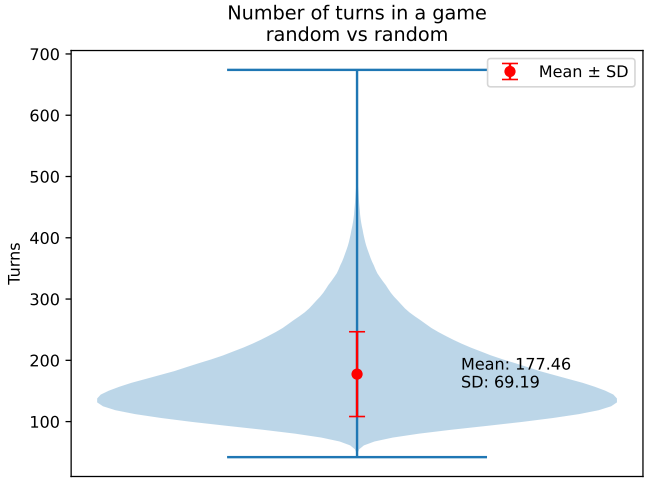


Figure 5.1: Plot with the number of turns in a game

5.2.2 Branching Factor

The branching factor measures the number of legal actions available to the player to move. It is one of the most important indicators of search complexity, since many tree-search algorithms have a cost that grows exponentially with the branching factor.

The average branching factor observed during the simulations is shown in 5.2. Across the simulated games, Tricolor has an average branching factor of approximately 47, with a standard deviation of 9.

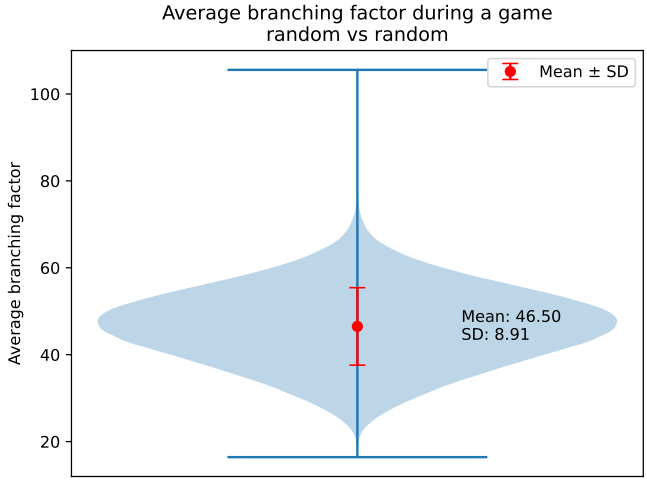
This value is high. It is larger than the classical average branching factor of Chess, often estimated around 35 [1]. In the initial position of Tricolor, the number of legal actions is particularly high because many pieces are still available and many stacks can move in several directions. As the game progresses, pieces are captured or imprisoned, which tends to reduce the number of available actions.

The observed standard deviation reflects the diversity of game trajectories. Games in which captures occur early tend to have lower branching factors later on, because fewer active stacks remain available. Conversely, games in which players avoid major confrontations for longer tend to preserve larger numbers of legal actions.

The combination of a high branching factor and long game duration is one of the main reasons why Tricolor is computationally challenging. Even relatively shallow search depths may already involve a very large number of possible continuations.

5.2.3 Game Tree Complexity

Game tree complexity estimates the number of possible games that can be played from the initial position until termination. Computing this value exactly is infeasible for Tricolor, as it would require enumerating all legal continuations of the game. A common approximation



5

Figure 5.2: Plot with average branching factor in a game

is:

$$GTC \approx b^d$$

,

where b is the average branching factor and d is the average game depth [1]. Using the empirical values obtained from random simulations,

$$b \approx 47 \text{ and } d \approx 177,$$

we obtain:

$$GTC \approx b^d = 47^{177} \approx 47^{1.2 \times 148} = (47^{1.2})^{148} \approx 100^{148}$$

Taking the base-10 logarithm gives approximately:

$$log_{10}(GTC) \approx 296.$$

Thus, the game tree complexity of Tricolor is roughly:

$$GTC \approx 10^{296}$$

This value should be interpreted as a rough estimate rather than an exact result. It depends on the average branching factor and average game length observed under random play, which may differ from values obtained under stronger play. Nevertheless, the estimate clearly indicates that Tricolor has an extremely large game tree.

This has direct consequences for AI. Exhaustive search is not realistic, and even optimized search methods can only examine a small fraction of the possible continuations. Therefore, strong play requires selective search, effective pruning, or heuristic evaluation.

5.2.4 State Space Complexity

State space complexity estimates the number of distinct positions that may occur in the game. Computing the exact number of reachable states is difficult, because not every theoretical placement of pieces corresponds to a legal position reachable from the initial state. Instead, this thesis considers an upper bound.

The board contains 61 tiles. Each player starts with 18 pieces, and pieces may be stacked on any tile. Since captured pieces are removed from the game, a player may have between 0 and 18 pieces remaining on the board.

For a given number w of white pieces, the number of ways to distribute them over 61 tiles is given by the stars-and-bars formula:

$$\binom{60+w}{w}$$

Similarly, for b black pieces, the number of possible distributions is:

$$\binom{60+b}{b}$$

Summing over all possible remaining numbers of white and black pieces gives:

$$\sum_{w=0}^{w=18}\sum_{b=0}^{b=18}\binom{60+w}{w}\binom{60+b}{b} = \sum_{w=0}^{w=18}\binom{60+w}{w}\cdot\sum_{b=0}^{b=18}\binom{60+b}{b}$$

Using the identity

$$\sum_{k=0}^{k=n}\binom{x+k}{k} = \binom{x+n+1}{n}$$

this becomes:

$$\binom{60+18+1}{18}\cdot\binom{60+18+1}{18} = \binom{79}{18}^2$$

This distribution does not yet encode stack ownership in positions where both players have pieces on the same tile. Since at most 18 mixed stacks can exist, multiplying by 2^{18} gives an upper bound for ownership possibilities. Finally, the player to move adds a factor of 2. Therefore:

$$\binom{79}{18}^2\cdot 2^{18}\cdot 2 = \binom{79}{18}^2\cdot 2^{19}$$

This gives an upper bound of approximately:

$$UB(SSC) = \binom{79}{18}^2 \cdot 2^{19} \approx (2 \cdot 10^{15})^2 \cdot 2^{19} = 10^{30} \cdot 2^{21} \approx 10^{30} \cdot 2^{3.3 \cdot 6.4} = 10^{30} \cdot (2^{3.3})^{6.4} \approx 10^{36}$$

Therefore, the state space complexity of Tricolor is bounded above by roughly:

$$UB(SSC) \approx 10^{36}$$

This value is high, although still below the classical state-space estimates often given for Chess, which are around 10^{47} [1]. The result highlights an interesting distinction: Tricolor has a very large game tree because games are long and branching factors are high, but its state space remains smaller than that of some more widely studied games.

This difference is important. A large game tree makes direct search difficult, while a comparatively smaller state space suggests that future work could potentially investigate transposition tables, endgame databases, or retrograde analysis for restricted subsets of the game.

5

5.3 Game Outcome and Balance

The final outcome of each simulated game was recorded as a win for Player 1, a win for Player 2, or a draw. Since the initial board setup is symmetric, one important question is whether the first player has a measurable advantage under random play.

The results are shown in 5.3. Player 1 wins approximately 49.4% of the games, while Player 2 wins approximately 48.9%. Draws account for approximately 1.8% of the simulated games.

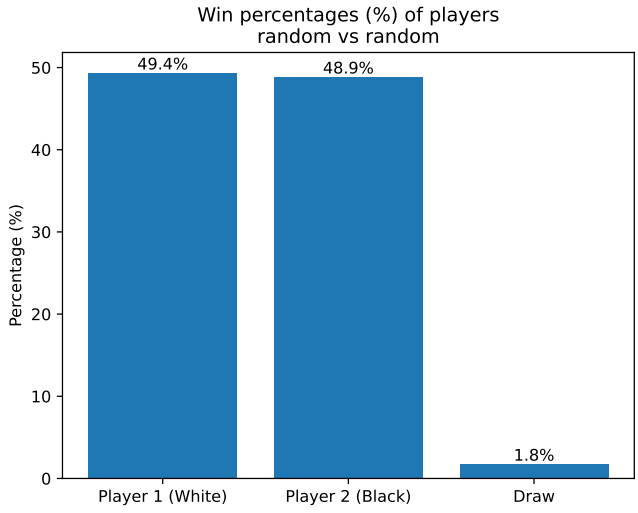


Figure 5.3: Plot with the win percentage of each player and draws

The difference between the win rates of the two players is only 0.5 percentage points. Given the size of the sample, this suggests that there is no substantial first-player advantage under random play. This is consistent with the symmetric nature of the initial position.

The draw rate is very low. This indicates that, even when both players move randomly, most games eventually lead to a decisive outcome. The low draw rate is partly explained by the capture and imprisonment mechanics: once a player begins to lose control over stacks, their mobility may decrease rapidly, eventually leaving them with no legal moves.

However, these results should not be generalized too quickly to expert play. A low draw rate under random play does not necessarily imply a low draw rate between strong agents. Strong players may be better at preserving material, avoiding forced weaknesses, or maintaining defensive structures. The comparison with stronger AI agents is therefore revisited later in the thesis.

5.4 Gameplay and Aesthetic Metrics

In addition to classical complexity measures, this thesis also studies several gameplay-related metrics inspired by previous work on game design and general game concepts [2] [24]. These metrics are useful because they describe how a game evolves dynamically, rather than only how large its search space is.

To compute several of these metrics, it is necessary to define an evaluation function for a position.

5.4.1 Position Evaluation

The purpose of the position evaluation function is to estimate whether a given position favors Player 1 or Player 2. The score is normalized in the interval $[-1, 1]$, where:

- 0 represents an approximately balanced position
- a positive score indicates an advantage for Player 1
- a negative score indicates an advantage for Player 2

The evaluation is based on the total controlled power of each player. Let w_s be the total score of the white player and b_s the total score of the black player. The position evaluation is:

$$eval = \frac{w_s - b_s}{w_s + b_s}$$

The value w_s is computed as:

$$w_s = \sum_{i=0}^{i=60} c_i * w_i$$

where c_i is the value of the color of tile i, and w_i is the number of white pieces controlled on that tile. The value b_s is computed similarly for black pieces.

This evaluation function captures two intuitive ideas. First, a player with more controlled pieces generally has more mobility and more opportunities to attack. Second,

controlling pieces on stronger tiles is valuable because tile colors affect attack and defense power.

This evaluation is intentionally simple. It is not meant to represent perfect strategic understanding of Tricolor. Instead, it provides a consistent numerical basis for measuring game dynamics such as lead changes, decisiveness, and drama.

5.4.2 Narrowness

Narrowness measures how many actions are worth considering in a given position. In this thesis, an action is considered relevant if it improves or maintains the current evaluation score for the player to move.

The average narrowness observed in the simulations is shown in 5.4. The mean value is approximately 25, with a standard deviation of 5.

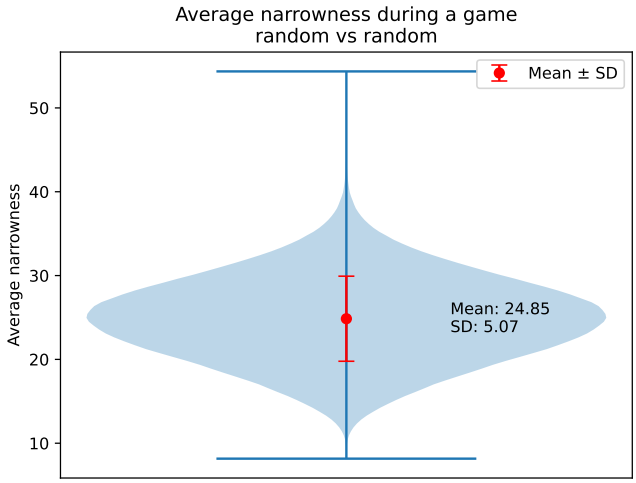


Figure 5.4: Plot with the average narrowness during a game

Since the average branching factor is approximately 47, this means that roughly half of the available moves preserve or improve the evaluation according to the simple evaluation function. This suggests that Tricolor often presents players with a relatively large number of plausible choices.

From an AI perspective, this is important. If only a small number of moves were relevant in most positions, move ordering or pruning would be easier. Instead, Tricolor frequently offers many moves that appear reasonable according to material and positional criteria. This increases the difficulty of selecting strong actions.

The interpretation of narrowness depends strongly on the evaluation function. A stronger heuristic might classify fewer moves as genuinely relevant. Nevertheless, the result indicates that Tricolor is not a game where random positions usually contain only one or two obvious candidate moves.

5.4.3 Decisiveness

Decisiveness measures the proportion of the game completed before the eventual winner obtains an advantage that is maintained until the end. This metric is computed only for games with a winner; drawn games are excluded.

The distribution of decisiveness values is shown in 5.5. The average decisiveness is approximately 70%, with a standard deviation of 22%.

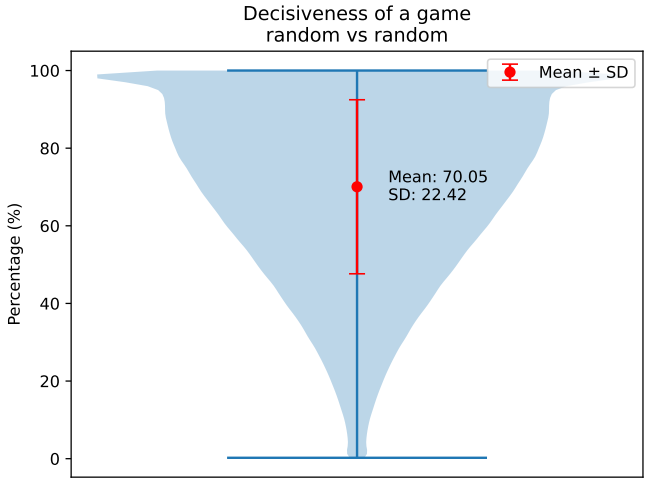


Figure 5.5: Plot with the game decisiveness

This relatively high value means that, on average, under random play, the eventual winner often does not establish a stable advantage until late in the game. In many games, the evaluation may fluctuate for a long time before one player finally obtains a decisive lead.

This result suggests that Tricolor has a high degree of uncertainty under random play. The outcome often remains unclear until the final phase of the game. This is consistent with the presence of imprisonment mechanics: pieces that appear lost may later be recovered if control of a stack changes.

From a gameplay perspective, this can be seen as an interesting property. Games in which the winner becomes obvious too early may feel less engaging, while games in which the evaluation remains unstable can maintain tension until the end.

5.4.4 Stability

Stability measures how often the lead changes during a game. In this thesis, a lead change occurs when the sign of the evaluation score changes from positive to negative or from negative to positive.

The percentage of lead-changing turns is shown in 5.6. The average value is approximately 16%, with a standard deviation of 7%.

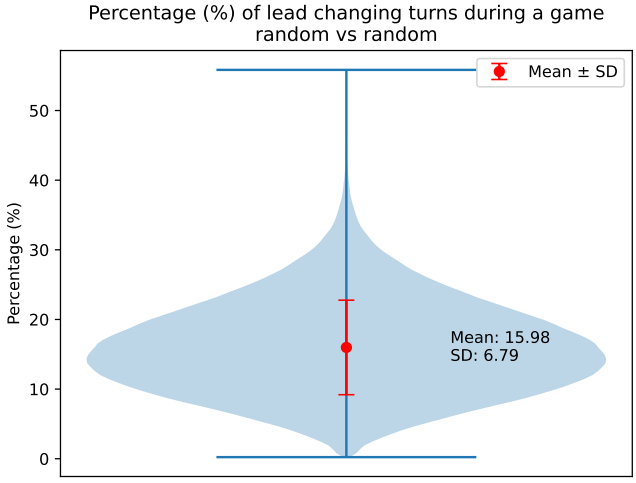


Figure 5.6: Plot with the percentage of lead changing turns during a game

This means that in a typical random game, the apparent leader changes several times. This is consistent with the high decisiveness value reported above: if the final winner only stabilizes their advantage late in the game, then the lead is expected to fluctuate during earlier phases.

The relatively low stability of Tricolor under random play can be explained by the game's mechanics. A single capture or imprisonment may significantly modify the evaluation, especially if it changes control of a large stack or frees imprisoned pieces. The forced-move rule may also cause sudden reversals by compelling players to enter tactical sequences.

5.4.5 Drama

Drama measures the average absolute change in evaluation between consecutive positions. It captures how abruptly the apparent advantage changes during play.

The distribution of drama values is shown in 5.7. The average drama is approximately 0.07, with a standard deviation of 0.02.

Since the evaluation score lies between -1 and 1, an average change of 0.07 is moderate. Most individual moves do not radically transform the position. However, this does not mean that the game lacks dramatic moments. Certain tactical sequences, especially those involving imprisonment and later recovery of pieces, can produce large swings in evaluation.

Thus, the game appears to combine gradual positional evolution with occasional high-impact tactical events. This is a meaningful property for both strategic analysis and AI design: agents must evaluate not only immediate material changes, but also the possibility of future reversals caused by stack control.

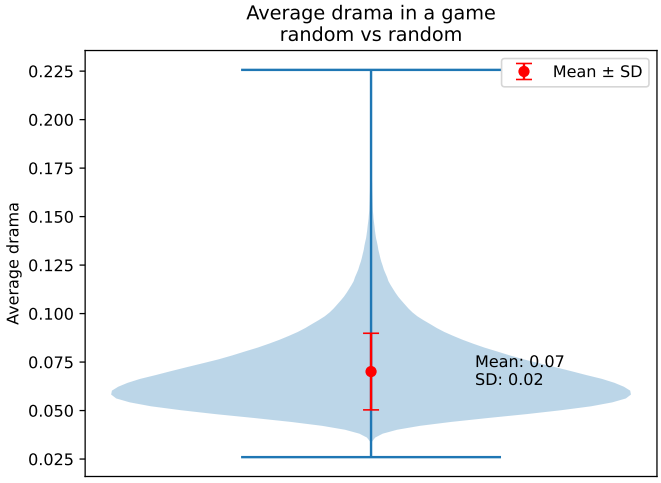


Figure 5.7: Plot with the average drama in a game

5.4.6 Example Game Evaluations

To illustrate the evolution of position evaluations during play, two example games between Random Agents are shown in 8.8 and 8.9.

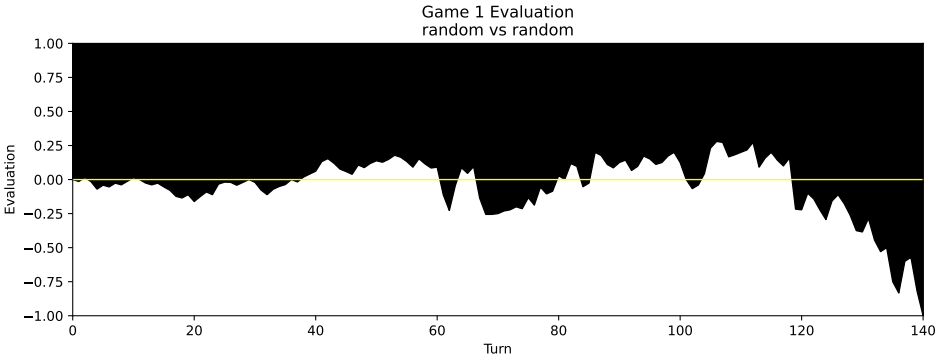


Figure 5.8: Plot with game 1 evaluation

In the first example, Player 2 eventually wins. The evaluation curve crosses the equality line several times, indicating multiple lead changes. Later, Player 1 obtains a favorable position, but Player 2 manages to recover and eventually maintain the advantage until victory. This example illustrates how Tricolor positions can remain unstable for many turns before a decisive advantage appears.

In the second example, Player 1 wins. During the first part of the game, the evaluation

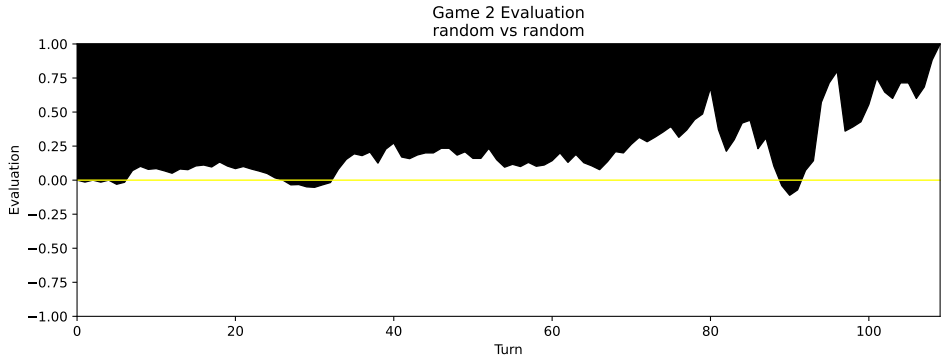


Figure 5.9: Plot with game 2 evaluation

remains relatively close to equality. The final advantage is established only late in the game. This example is consistent with the high decisiveness values measured across the full simulation set.

These examples show that the aggregate metrics are not merely abstract quantities. They correspond to concrete gameplay patterns: fluctuating evaluations, late stabilization of advantage, and occasional dramatic shifts caused by tactical interactions.

5.5 Additional Metrics

Several additional metrics were also computed and are reported in Appendix A. These include:

- *Decision Moves*, measuring the percentage of turns in which more than one legal move is available;
- *State Repetition*, measuring how often positions repeat during a game;
- *Decision Factor*, measuring the average number of legal actions when forced single-move turns are excluded;
- *Board Sites Occupied*, measuring how many tiles are occupied at least once during a game;
- *Move Distance*, measuring the average distance travelled by moved pieces;
- *Piece Number*, measuring the average number of pieces remaining on the board during a game;
- *Lead Change Moves*, measuring how many available actions would change the current leader.

These metrics provide additional information about the structure and dynamics of Tricolor. For example, the high percentage of decision moves indicates that most turns

offer meaningful choice, while the low number of repeated states suggests that games rarely cycle extensively under random play. The board occupation and movement-distance metrics also show that games tend to use a substantial portion of the board, rather than remaining confined to the initial setup area.

The purpose of including these metrics in the appendix is to provide a broader empirical profile of the game without overloading the main analysis. They may be useful for future comparisons with other games or with alternative versions of Tricolor.

5.6 Discussion

The results presented in this chapter show that Tricolor is a computationally demanding game. Its average branching factor is high, its games are relatively long, and its estimated game tree complexity is extremely large. These properties make exhaustive search infeasible and justify the need for selective AI methods.

At the same time, the estimated state space upper bound is not as large as the game tree complexity. This suggests that Tricolor may contain many different paths leading to overlapping or related positions. Future work could investigate the use of transposition tables or state-space exploration techniques to better understand this structure.

The outcome statistics suggest that the game is relatively balanced under random play, with no clear first-player advantage. The low draw rate indicates that the capture and imprisonment mechanics usually lead to decisive outcomes, even without strategic play.

The gameplay metrics reveal an additional layer of interest. Tricolor is not only computationally large; it is also dynamically unstable. Lead changes occur frequently, decisive advantages often appear late, and tactical interactions can produce meaningful reversals. These properties suggest that the game contains rich strategic possibilities despite its limited historical study.

The analysis also highlights the limitations of random simulations. Random agents are useful for estimating baseline properties, but they do not represent strong play. Some measured values, such as game duration, branching factor, draw rate, and decisiveness, may change significantly when stronger agents are used. This is investigated later in the thesis through comparisons between Alpha-Beta and Monte Carlo Tree Search agents.

Overall, this chapter establishes Tricolor as a deep, high-branching, strategically dynamic combinatorial game. These observations motivate the next stage of the thesis: identifying strategic principles and heuristic evaluations that can guide AI agents in such a large search space.

6

Strategic Analysis and Heuristic Design

6.1 Need for Heuristics

The previous chapter showed that Tricolor has a large branching factor, long game durations, and a very large estimated game tree complexity. As a consequence, search-based agents cannot usually explore complete games from a given position. They therefore require heuristic evaluation functions to estimate the quality of non-terminal positions.

Heuristic design is a central problem in game-playing AI. In general game playing, different games require different evaluation criteria, and recent work has studied how suitable heuristics can be predicted from formal game descriptions [25]. In the case of Tricolor, no existing strategic theory is available. The heuristic used in this thesis is therefore designed manually from observations of the rules and from experimental play.

A good heuristic for Tricolor should reflect the main structural features of the game: control of pieces, positional strength, stack management, and the tactical consequences of forced capture or imprisonment moves. The following sections describe the main strategic observations used to construct such an evaluation function.

6.2 First Heuristic

The objective of Tricolor is to leave the opponent without any legal moves during their turn. This is achieved by capturing or imprisoning all of the opponent's pieces. A natural baseline heuristic for evaluating a game state is therefore to compare the number of controlled pieces on the board for each player, as this provides a coarse approximation of relative material advantage.

6.3 Second Heuristic

For each player, capturing and imprisoning the opponent's pieces, as well as defending their own, is strongly influenced by the positioning of stacks on powerful tiles. Consequently, a second heuristic consists of comparing the total accumulated strength of all controlled stacks on the board between the two players.

6.4 Third Heuristic

At first glance, tall stacks positioned on red tiles may appear impossible to capture or imprison due to their high power. However, the forced-move rule in Tricolor can compel players to dislocate such stacks to weaker tiles, thereby significantly increasing their vulnerability.

This insight motivates a new strategy, referred to in this paper as *single-piece baiting* attack.

To better understand this strategy, we consider the example presented in Scenario 6.1:

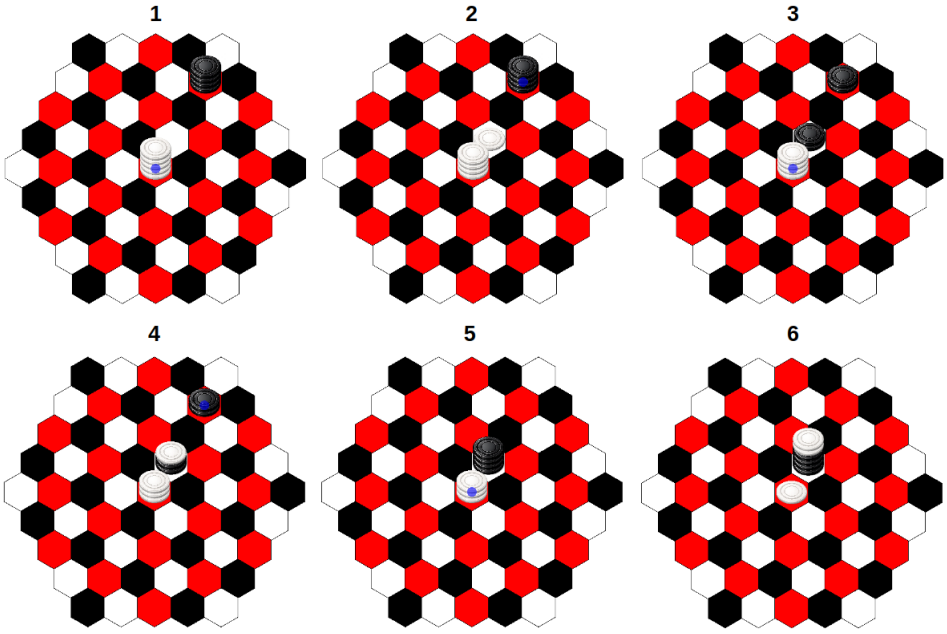


Figure 6.1: Figure displaying how the *single-piece baiting* strategy used by the white player defeats the powerful stack of the black player.

This scenario illustrates how Player 1 forces Player 2 to dislocate their stack onto a white tile by moving a single piece at each turn. Since Player 2 is located further from the white tile, they are required to dislocate at least two pieces per turn. In the long run, this leads to Player 2 losing more pieces than Player 1, resulting in a win for Player 1.[1]

An important question that arises is whether a smaller stack can defeat a larger stack using a *single-piece baiting* attack.

In the following paragraph, it will be shown that the answer to this question is negative.

Proof:

[1] After the final move, Player 1 still has one piece remaining on the initial tile, indicating that this strategy also applies when both players start with the same number of pieces on the initial tiles.

First, it is important to specify that only scenarios in which both players have stacks on red tiles are relevant. This is because strong players typically aim to place their stacks onto red tiles as quickly as possible. Furthermore, we restrict our attention to positions in which the bait tile is white. Using a black or red tile as bait would advantage the defender by providing a more strongly supported stack, thereby forcing the attacker to commit additional pieces in order to sustain the attack.

From the geometry of the board, it can be observed that for any given tile, all tiles at distance three share the same color, as shown in 6.2. This implies that a stack on a red tile cannot be attacked using a *single-piece baiting* strategy with a bait tile located at distance three from the defender. In the most favorable case for the attacker, the white bait tile is located at distance two from the defender, as illustrated in the previous example 6.1.

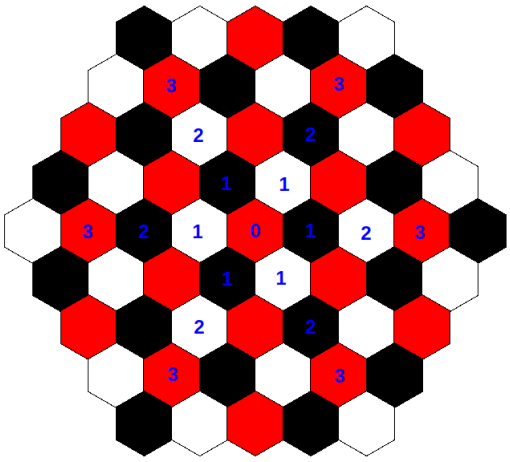


Figure 6.2: Figure showing that all the tiles at distance 3 of the central red tile are also red.

Now that the relevant scenarios for the single-piece baiting strategy have been established, we show that if the attacker has fewer pieces than the defender, the attack can never succeed. The flowchart in 6.3 supports this claim by enumerating all possible moves available to the attacker.

Finally, this leads to the design of a third heuristic, which consists of prioritizing the concentration of pieces onto a minimal number of tiles. This is based on the observation that a small number of large stacks is preferable to a larger number of small stacks. As shown previously, a large stack on a red tile cannot be the victim of a *single-piece baiting* attack from a smaller stack, whereas the opposite situation does not hold.

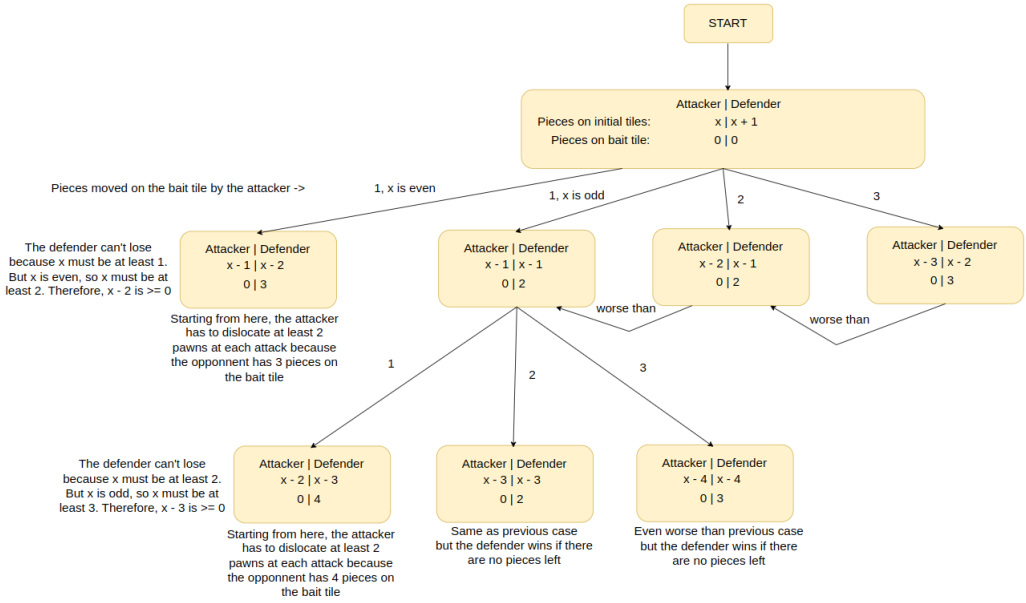


Figure 6.3: Flowchart illustrating all possible move sequences during a *single-piece baiting* attack.

6.5 Combined Heuristic and Limitations

The heuristic evaluation used in this thesis combines the previous observations in a prioritized way. The first priority is the comparison of controlled material, since control of pieces directly affects mobility and the possibility of winning. The second priority is positional power, which rewards control of stronger tiles and more powerful stacks. Finally, the evaluation favors positions where pieces are grouped into stronger stacks rather than being spread across many weak stacks.

This prioritization is important. If positional power were weighted too strongly, an agent might sacrifice too many pieces only to occupy stronger tiles. Such behavior may improve the position locally, but it can be strategically poor if it reduces long-term control and mobility. Giving priority to material control helps avoid this problem.

The proposed heuristic remains handcrafted and therefore limited. It does not capture all tactical motifs of Tricolor, and its weights are not learned automatically from data. Moreover, some strategic features may only become visible through stronger self-play or deeper search. Nevertheless, the heuristic provides a useful first evaluation function for the Alpha-Beta agent introduced in the next chapter.

Future work could investigate automatic heuristic tuning or learning-based evaluation functions. This would be consistent with previous work on heuristic prediction in general game playing, where game descriptions are used to estimate which heuristic features are likely to perform well [25]. In the case of Tricolor, such approaches could eventually help refine or replace the manually designed evaluation function.

7

AI Agents

This chapter presents the artificial intelligence agents developed for Tricolor. The focus is placed on search-based approaches capable of making strategic decisions from the current game state. The Random Agent introduced in Chapter 4 is used only as a baseline in later experiments and is therefore not described again here.

Two main agents are implemented and analyzed in this thesis: an Alpha-Beta Agent and a Monte Carlo Tree Search Agent. Both operate under the same decision-time constraint and use the game engine described in Chapter 4 to generate legal moves and simulate resulting positions. An additional Expert Iteration approach is also considered as ongoing work.

7.1 Alpha-Beta Agent

The Alpha-Beta Agent is based on the minimax search principle with Alpha-Beta pruning [9]. Since Tricolor is a deterministic two-player game with perfect information, it can be represented naturally as an adversarial search problem. From a given position, the agent explores possible continuations by alternating between maximizing and minimizing nodes, corresponding to the two players.

Alpha-Beta pruning improves minimax search by avoiding the exploration of branches that cannot influence the final decision. This is particularly important in Tricolor because the game has a high branching factor, as shown in Chapter 5. Without pruning, even shallow searches become computationally expensive.

Since complete search is infeasible, the agent uses depth-limited search. When the maximum depth is reached and the position is not terminal, the heuristic evaluation function described in Chapter 6 is used to estimate the quality of the position. This evaluation combines material control, positional power, and stack-related considerations.

The implementation also uses iterative deepening [9]. Instead of fixing a single depth in advance, the agent searches first at depth 1, then depth 2, and so on, until the available time for the move is exhausted. This ensures that the agent always has a valid move available, while allowing deeper searches in positions where the branching factor is lower.

A time limit of one second per move is used in the experiments. This constraint makes the comparison between agents fair and reflects the practical need for agents to make decisions within a limited amount of time.

Move ordering is also used to improve pruning efficiency [9]. Before exploring the children of a position, actions are sorted according to their heuristic evaluation. Promising moves are searched first, increasing the chance that Alpha-Beta pruning can discard weaker branches earlier. This optimization is especially useful in a game such as Tricolor, where many legal actions may be available at each turn.

Overall, the Alpha-Beta Agent represents a heuristic-search approach. Its performance depends strongly on the quality of the evaluation function and on the ability of move ordering to guide the search toward relevant branches.

7.2 Monte Carlo Tree Search Agent

The Monte Carlo Tree Search Agent uses simulations to estimate the value of available moves [9]. Unlike Alpha-Beta search, MCTS does not require a handcrafted evaluation function at leaf nodes. Instead, it repeatedly explores the game tree by running simulated games and using their outcomes to guide future exploration.

The implementation follows the standard four-step MCTS procedure.

First, during the selection phase, the algorithm starts from the root node and recursively selects child nodes according to the UCT formula:

$$\frac{w_i}{n_i} + c \cdot \sqrt{\frac{\ln N}{n_i}}$$

where

- w_i is the number of wins obtained through child i
- n_i is the number of visits of child i
- N is the number of visits of the parent node
- c is the exploration constant. In this implementation, $c = \sqrt{2}$, a standard value commonly used in UCT-based search [26].

Second, during the expansion phase, one previously unexplored child of the selected node is added to the tree.

Third, during the simulation phase, a game is played from the expanded node until a terminal state is reached. In the current implementation, simulations are performed using random play.

Finally, during the backpropagation phase, the result of the simulated game is propagated back through the nodes visited during selection. Visit counts and win counts are updated accordingly.

As with the Alpha-Beta Agent, the MCTS Agent is given a time limit of one second per move. This ensures that both agents operate under comparable computational constraints.

Tricolor presents several challenges for MCTS. The average game duration is relatively long, which makes random simulations expensive. In addition, the branching factor is high,

meaning that the algorithm must distribute its simulations among many possible actions. Finally, random playouts may provide noisy estimates in a game where stack control and forced tactical sequences can create delayed consequences. These factors can reduce the effectiveness of MCTS when used without domain-specific enhancements.

7.3 Justification of Agent Design Choices

The Alpha-Beta and MCTS agents were selected because they represent two classical and complementary approaches to game-playing AI.

The Alpha-Beta Agent relies on explicit heuristic evaluation. It is expected to perform well when the evaluation function captures important strategic properties of the game and when pruning can reduce the effective branching factor. This makes it particularly suitable for testing the heuristic ideas developed in 6.

The MCTS Agent follows a simulation-based approach. It is attractive because it can operate without a handcrafted evaluation function, but its performance depends strongly on the number and quality of simulations. In Tricolor, long games, high branching factors, and delayed consequences caused by stack interactions make pure random rollouts potentially unreliable.

The Expert Iteration [27] approach remains a promising extension because it could combine search and learning. However, such methods require additional implementation, training data generation, model design, and evaluation. They are therefore treated as ongoing work rather than as the main focus of this thesis.

The next chapter evaluates the implemented agents experimentally and compares their performance in simulated games.

8

Experimental Evaluation

This chapter evaluates the performance of the artificial intelligence agents introduced in Chapter 7. The objective is to compare their playing strength, analyze their behavior, and relate the experimental results to the structural properties of Tricolor identified in Chapter 5.

The source code of the implementation is publicly available on GitHub: https://github.com/AndreiBourceanu/TFE_UCL/tree/main/Tricolor_framework. All experiments reported in this chapter were run on an Intel Core i5-8250U CPU with 8 GB of memory, using the Ubuntu 22.04.5 LTS operating system and compiled with g++ (GCC) 11.4.0. Unless otherwise stated, each advanced agent was given a decision time of one second per move. This information is provided for reproducibility, since the performance of search-based agents depends directly on the amount of computation available during move selection.

8.1 Experimental Protocol

The experiments compare the agents implemented in this thesis under controlled conditions. The Random Agent is used as a baseline, while the Alpha-Beta Agent and the Monte Carlo Tree Search Agent are evaluated as the main AI approaches.

Each advanced agent is given a fixed decision time of one second per move. This constraint ensures that the comparison between agents is fair and that neither agent benefits from additional computation time. The same game engine, move generator, and terminal-state detection mechanisms are used for all experiments.

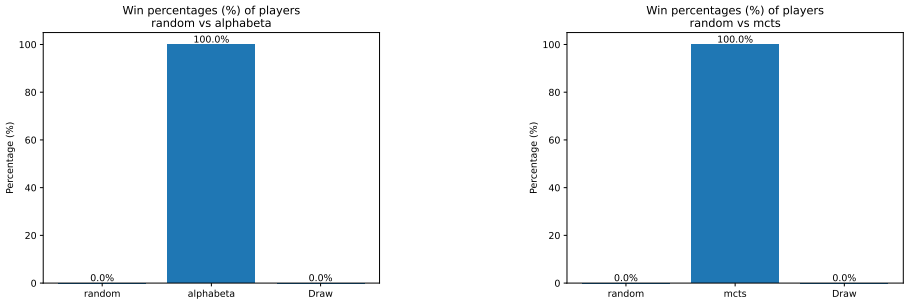
For each pair of distinct agents, 100 games are simulated. To reduce possible color bias, each agent plays 50 games as Player 1 and 50 games as Player 2. This is particularly important because Player 1 always moves first. Although the initial position is symmetric, alternating colors makes the comparison more robust.

The main performance metric is the final outcome of each game: win, loss, or draw. In addition, several game metrics are recomputed for games involving advanced agents, including game duration, branching factor, decisiveness, stability, and drama. These values are compared with the random-play baseline studied in Chapter 5.

The number of games is smaller than in the random simulations of Chapter 5 because advanced agents are much more expensive to run. Since each move may take up to one second and Tricolor games can last many turns, simulating thousands or millions of games between advanced agents would require a substantial amount of computation time.

8.2 Baseline Results Against the Random Agent

As a first validation step, the Alpha-Beta Agent and the Monte Carlo Tree Search Agent are evaluated against the Random Agent. The results are shown in 8.1.



(a) Random vs Alpha-Beta

(b) Random vs MCTS

Figure 8.1: Plots displaying the win percentage of the Random Agent vs the Alpha-Beta Agent and the MCTS Agent.

Both advanced agents win all games against the Random Agent. The Alpha-Beta Agent obtains a win rate of 100%, and the MCTS Agent also obtains a win rate of 100%.

These results are expected, but they are still useful. They confirm that both agents are able to exploit the rules of Tricolor more effectively than random play. They also indicate that the implementation of the advanced agents is functional: even without comparing them to each other, both clearly outperform the baseline.

However, these results should not be interpreted as evidence that both agents play strongly in an absolute sense. Defeating a Random Agent only shows that the agents can make better-than-random decisions. A more informative comparison is obtained by evaluating the two advanced agents directly against each other.

8.3 Alpha-Beta versus Monte Carlo Tree Search

The main experiment compares the Alpha-Beta Agent with the Monte Carlo Tree Search Agent. The results are shown in 8.2.

The Alpha-Beta Agent wins 96% of the games, while the MCTS Agent wins only 1%. The remaining 3% of games end in draws. This result indicates a clear advantage for the Alpha-Beta Agent under the experimental conditions used in this thesis.

Several factors can explain this result.

First, the Alpha-Beta Agent benefits from the heuristic evaluation function introduced in Chapter 6. This heuristic incorporates domain-specific knowledge about material control,

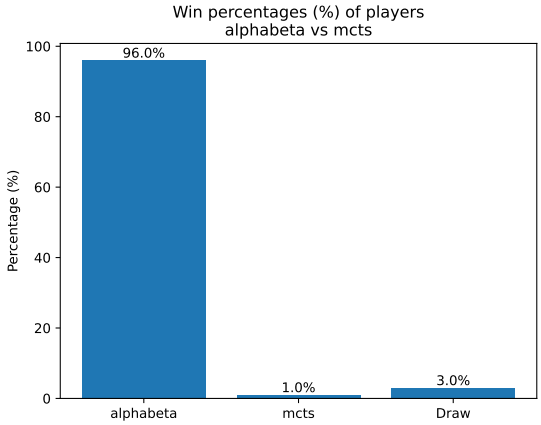


Figure 8.2: Plot displaying the win percentage of the Alpha-Beta Agent vs the MCTS Agent.

positional power, and stack concentration. As a result, the agent can evaluate non-terminal positions without needing to simulate complete games.

Second, the game properties identified in Chapter 5 are unfavorable to a basic MCTS implementation. Tricolor has a high branching factor and long game durations. This means that, within a one-second time limit, MCTS must distribute a limited number of simulations across many possible actions, and each simulation may be relatively expensive.

Third, the simulations used by the MCTS Agent are based on random play. In a game such as Tricolor, random rollouts may provide noisy evaluations because important strategic consequences can appear only after long tactical sequences. Stack control, imprisonment, and forced moves may produce delayed effects that are difficult to estimate reliably through purely random simulations.

By contrast, Alpha-Beta search focuses on a more structured exploration of the game tree and uses the heuristic to evaluate positions reached at limited depth. In this experimental setting, this combination appears better suited to Tricolor than basic MCTS with random rollouts.

8.4 Game Metrics using AI Agents

In addition to win rates, several metrics introduced in Chapter 5 were recomputed for games played between the Alpha-Beta Agent and the MCTS Agent. The objective is not to redefine these metrics, but to compare the dynamics of games played by advanced agents with those observed in random play.

Games between Alpha-Beta and MCTS are significantly shorter than games between two Random Agents. The average duration decreases from approximately 177 turns in random games to 77 turns in Alpha-Beta versus MCTS games 8.3. This difference suggests that the Alpha-Beta Agent is able to exploit weaknesses in the MCTS Agent quickly, leading to faster decisive outcomes.

The average branching factor follows the opposite trend. It increases from approx-

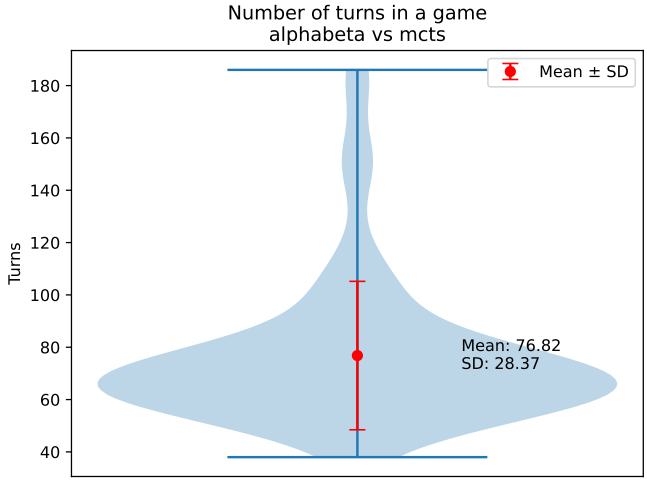


Figure 8.3: Plot with the number of turns in a game

imately 47 in random games to 77 in Alpha-Beta versus MCTS games 8.4. This can be explained by the fact that these games are shorter and spend less time in reduced-material late-game positions. In particular, the Alpha-Beta Agent tends to preserve and organize material more effectively during the early stages of the game, so more pieces remain active for a larger proportion of the game.

The decisiveness value is considerably lower, decreasing from approximately 70% in random games to 31% in Alpha-Beta versus MCTS games 8.5. This indicates that, according to the evaluation function, the eventual winner typically establishes a stable advantage relatively early in the game. Since Alpha-Beta wins the vast majority of games, this result suggests that the agent is often able to convert its superior decisions into a lasting positional advantage well before the endgame.

The stability metric also changes clearly. The percentage of lead-changing turns decreases from approximately 16% in random games to 5% in Alpha-Beta versus MCTS games 8.6. This indicates that games between advanced agents are much more stable according to the evaluation function. Once Alpha-Beta obtains an advantage, it tends to maintain it more consistently than a Random Agent would.

Drama also decreases slightly, from approximately 0.07 in random games to 0.05 in Alpha-Beta versus MCTS games 8.7. This suggests that evaluation changes are more gradual when advanced agents are involved. Random play often creates sudden tactical mistakes, while search-based play tends to avoid some of these abrupt reversals.

The two example games between the Alpha-Beta Agent and the MCTS Agent (Figures 8.8 and 8.9) illustrate the dominance of the Alpha-Beta Agent, which consistently maintains a visible positional advantage according to the evaluation function. As a result, lead changes almost never happen during the games. Moreover, the games end relatively quickly, after approximately 60 turns.

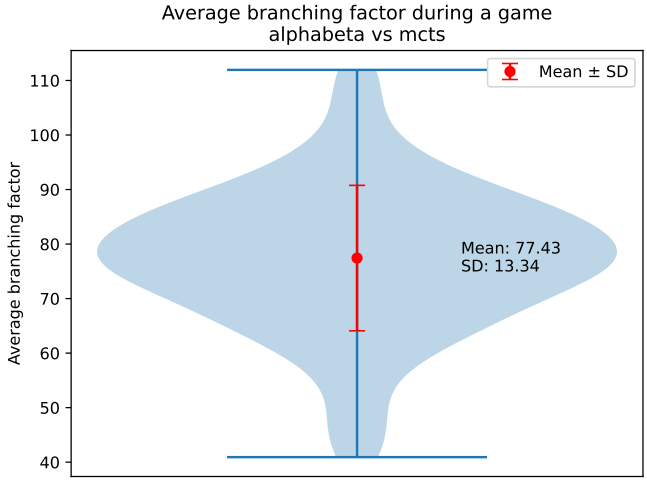


Figure 8.4: Plot with average branching factor in a game

Overall, these metrics show that the dynamics of Tricolor change substantially when moving from random play to search-based play. Games become shorter, more stable, and slightly less dramatic, while the average branching factor increases because the game spends less time in depleted late-game positions. In addition, the decisiveness metric suggests that Alpha-Beta asserts its dominance much earlier in the game, often establishing a stable advantage well before the final stages.

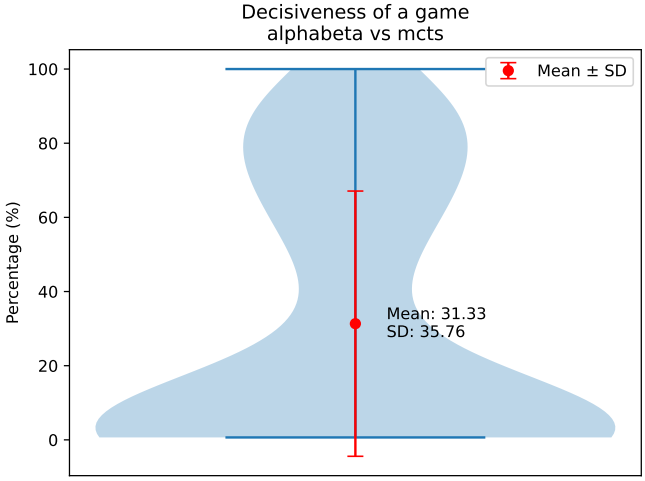


Figure 8.5: Plot with the game decisiveness

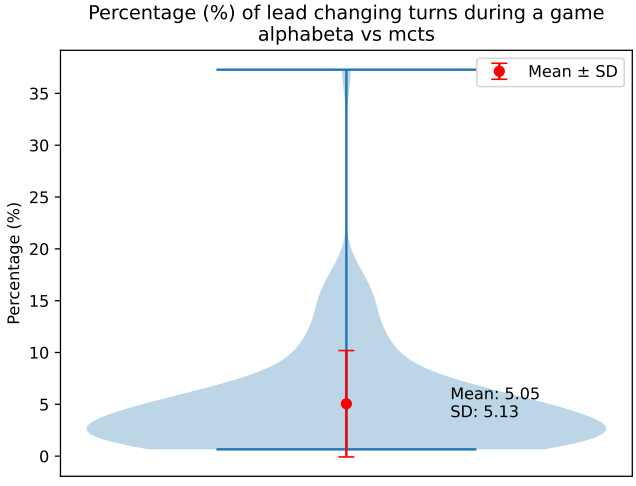


Figure 8.6: Plot with the percentage of lead changing turns during a game

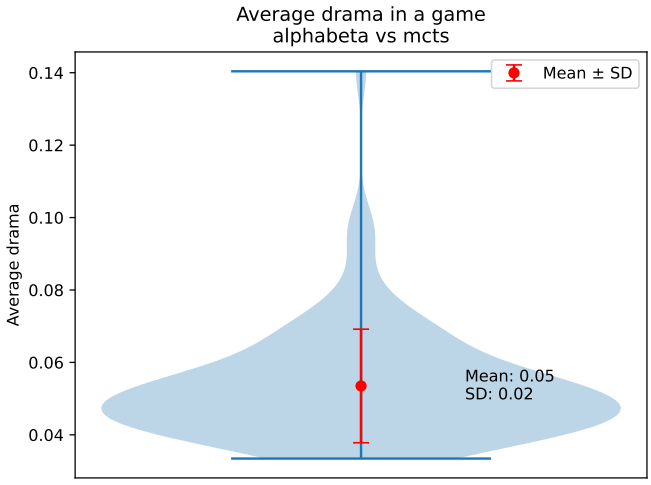


Figure 8.7: Plot with the average drama in a game

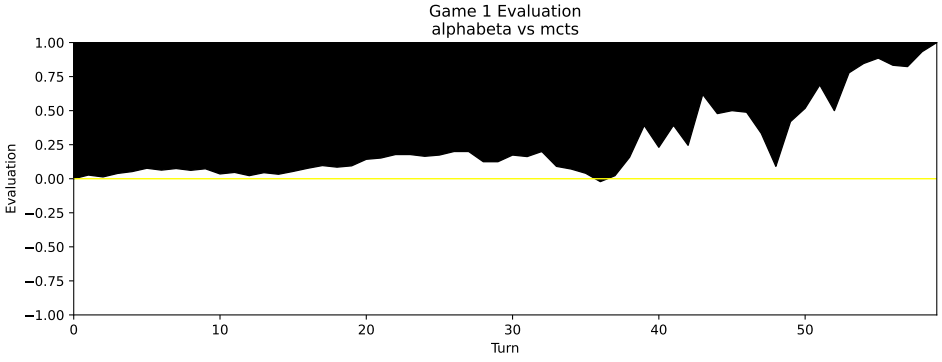


Figure 8.8: Plot with game 1 evaluation

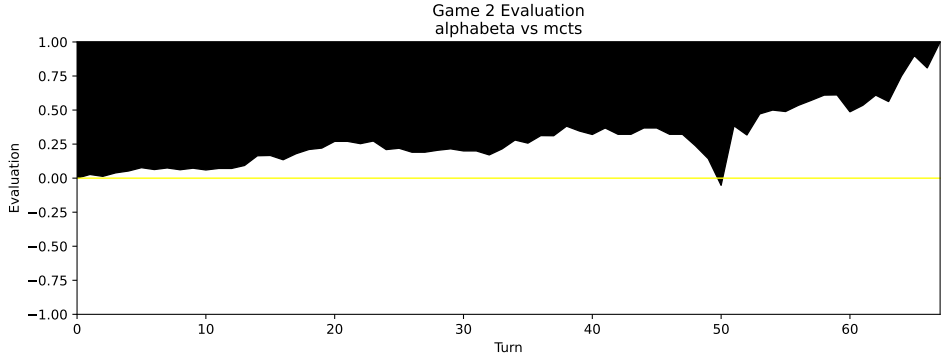


Figure 8.9: Plot with game 2 evaluation

8.5 Discussion

The experiments presented in this chapter show a clear difference between the two advanced agents. Both Alpha-Beta and Monte Carlo Tree Search are able to defeat the Random Agent consistently, but their direct comparison shows that the Alpha-Beta Agent is substantially stronger under the experimental conditions used in this thesis. This result is not only a comparison between two algorithms; it also provides insight into the computational nature of Tricolor.

The strong performance of the Alpha-Beta Agent suggests that the handcrafted heuristic introduced in Chapter 6 captures several important properties of the game. Material control, positional power, and stack concentration appear to provide useful information for evaluating non-terminal positions. Since Alpha-Beta cannot search to the end of the game, especially in early positions with many legal moves, its success depends heavily on the quality of this evaluation. The results therefore support the relevance of the strategic observations used to design the heuristic.

The weakness of the basic MCTS Agent can be explained by several properties of Tricolor. First, the game has a high branching factor, meaning that the agent must distribute its simulations across many possible actions. With a one-second time limit, many moves may receive only a limited number of visits. This makes it difficult for MCTS to distinguish reliably between promising and weak moves.

Second, Tricolor games can be long, especially when compared with the number of simulations that can be performed within a short decision time. Random rollouts may therefore be expensive and noisy. A simulation may take many turns before reaching a terminal result, and the final outcome may depend on random decisions made much later in the game. This weakens the relationship between the quality of the initial move and the result of the rollout.

Third, Tricolor contains delayed tactical consequences. A move that appears locally harmless may later force a player to split a strong stack, lose control of imprisoned pieces, or enter an unfavorable capture sequence. These consequences are difficult to evaluate through purely random playouts. In such situations, MCTS without a domain-specific rollout policy may fail to identify moves with strong long-term strategic implications.

The comparison between the agents also illustrates the importance of domain knowledge. Alpha-Beta is guided by a heuristic explicitly designed for Tricolor, while the MCTS Agent relies mostly on random simulations. The result therefore does not necessarily imply that MCTS is intrinsically unsuitable for Tricolor. Rather, it shows that a basic MCTS implementation is not sufficient to handle the game effectively under the tested conditions. Improved versions of MCTS could include heuristic rollouts, progressive bias, transposition tables, or learned value estimates.

The game-dynamics metrics reinforce this interpretation. Games between Alpha-Beta and MCTS are shorter, more stable, and less dramatic than games between Random Agents. The decisiveness metric suggests that once Alpha-Beta obtains an advantage, it is usually able to maintain and convert it efficiently. The increase in average branching factor in these games should not be interpreted as making the games harder for the agents throughout; rather, it reflects the fact that games end earlier and therefore spend less time in positions where many pieces have already been captured or imprisoned.

The experimental protocol also has limitations. Only 100 games were played for

each advanced-agent comparison, mainly because both agents require significantly more computation time than the Random Agent. Although the observed difference between Alpha-Beta and MCTS is large enough to support a clear conclusion, additional games would provide more precise statistical estimates. The experiments also use a fixed one-second time limit per move. Different time budgets could affect the results, especially for MCTS, which may benefit more directly from additional simulations.

Another limitation is that the agents were not extensively tuned. The Alpha-Beta Agent uses a manually designed heuristic, and the MCTS Agent uses a standard UCT configuration with random rollouts. Alternative parameter choices, different exploration constant.

9

Conclusion and Perspectives for the Study of Tricolor

9.1 Conclusions

This thesis presented the first systematic computational study of Tricolor, a board game invented by Maurice Kraitchik in 1930. Although the game occupies an interesting position in the history of abstract strategy games due to its early use of a hexagonal board and its original stack-based mechanics, it had previously received very little to no attention from the perspectives of artificial intelligence and computational game analysis.

The work began by formalizing the rules of Tricolor from historical sources and translating them into a precise computational model suitable for simulation and automated experimentation. Based on this formalization, an efficient game engine was implemented in C++, providing support for move generation, state manipulation, gameplay visualization, and AI-based search.

Large-scale simulations made it possible to characterize several structural properties of the game. The results showed that Tricolor combines relatively long games with a high branching factor, leading to an estimated game tree complexity comparable to that of many challenging combinatorial games. At the same time, the game exhibited balanced outcomes under random play and dynamic gameplay properties characterized by frequent lead changes and late-emerging decisive advantages.

The study also identified several strategic principles that emerge naturally from the mechanics of the game. Material control, positional power, and stack concentration were shown to play an important role in successful play and were incorporated into a heuristic evaluation function. Experimental results demonstrated that an Alpha-Beta agent guided by these strategic observations significantly outperformed a Monte Carlo Tree Search agent based on random simulations. These findings suggest that domain-specific knowledge is particularly valuable in Tricolor, where tactical consequences may be delayed and difficult to estimate through uninformed rollouts.

Overall, the results confirm that Tricolor is both strategically rich and computationally challenging. Beyond the performance of the implemented agents, the thesis establishes a foundation for future investigations of the game and provides tools that can support

further research in artificial intelligence, combinatorial game theory, and historical game studies.

9.2 Future Research Directions

Several directions for future work emerge from this study.

A first area concerns the development of stronger artificial intelligence agents. The Alpha-Beta agent implemented in this thesis relies on a manually designed heuristic, while the Monte Carlo Tree Search agent uses standard random rollouts. Future work could investigate more sophisticated search techniques, including transposition tables, improved move ordering, heuristic-guided simulations, or hybrid approaches combining search and learned evaluations. Learning-based methods are particularly promising. Self-play systems inspired by AlphaZero could potentially learn strategic patterns directly from generated games and discover positional concepts that are difficult to identify manually.

A second direction concerns the mathematical analysis of Tricolor. The complexity estimates presented in this thesis are based on empirical observations and theoretical upper bounds. More precise investigations could focus on the exact structure of the state space, the prevalence of transpositions, the existence of forced-win positions, or the identification of strategically important endgames. Such analyses could contribute to a deeper theoretical understanding of the game and its relationship to other combinatorial board games.

Future work could also extend the strategic study of Tricolor itself. The heuristics proposed in this thesis were derived from gameplay observations and limited experimentation. Stronger agents may reveal additional strategic principles, tactical motifs, or positional structures that remain undiscovered. Automated heuristic tuning and large-scale self-play experiments could help refine the current evaluation framework and provide a more complete description of expert-level play.

Finally, Tricolor offers interesting opportunities from a historical and cultural perspective. As one of the earliest known hexagonal board games, it represents an original contribution to the history of game design. The computational framework developed in this thesis could support future work aimed at preserving, documenting, and comparing historical games. Integration into general game systems such as Ludii would facilitate broader accessibility and enable direct comparison with other historical and modern board games. More generally, Tricolor illustrates how artificial intelligence can contribute not only to playing games, but also to understanding them as mathematical, historical, and cultural artifacts.

Taken together, these perspectives suggest that the present work should be viewed not as a definitive analysis of Tricolor, but as the starting point of a broader research program devoted to the study of this historically significant game.

A

Game metrics

This appendix presents the complete set of metrics computed for Tricolor, including those not discussed in Chapter 5 and Chapter 8.

For each metric, a definition is provided together with its source of inspiration, where applicable.

- *Duration* – measures the duration of a game in turns. Here, a turn is equivalent to a move made by any of the 2 players.

  Sources: [2] and [1].

- *Branching Factor* – measures how many possible actions any player can choose from, on average, during their turn.

  Source: [1].

- *Decision Moves* – measures what percentage of the moves made by any player during a game aren't forced. A move isn't forced if the player can choose between at least two actions during their turn.

  Source: [28].

- *State Repetition* – measures how many times the same states are repeated during a game.

  Source: [28].

- *Decision Factor* – measures how many possible actions any player can choose from, on average, during their turn, ignoring turns where only one move is possible.

  Source: [28].

- *Board Sites Occupied* – measures how many board tiles were occupied by any player at least once during the game.

  Source: [28].

A

- *Move Distance* – measures the average distance traveled by stacks during a game.

  Source: [28].

- *Piece Number* – measures the average number of pieces on the board during a game.

  Source: [28].

- *Game Outcome* – measures the percentage of games won by Player 1 (White), Player 2 (Black) and draws.

  Source: [2].

- *Narrowness* – measures the average narrowness of all the turns of a game. The narrowness of a turn represents the number of considerable actions a player can choose from during that turn. An action is considerable if it improves or maintains the current score of the position for the player to move.

  Sources: [24] and [28].

- *Decisiveness* – measures the proportion of the game completed before the eventual winner establishes a decisive advantage that is maintained until the end of the game. This metric is only computed when a game has a winning player; draws a discarded.

  Source: [24].

- *Lead Change* – measures the average number of possible actions during a turn that change the leader of the game. The leader of the game is the player who has a positional advantage.

  Source: [28].

- *Stability* – measures the percentage of moves that changed the leader during a game.

  Sources: [2] (referred to as *Uncertainty* in the paper) and [28].

- *Drama* – measures the average change in position evaluation scores throughout a game.

  Sources: [2], [24] and [28].

A.1 Random Agent vs Random Agent simulations

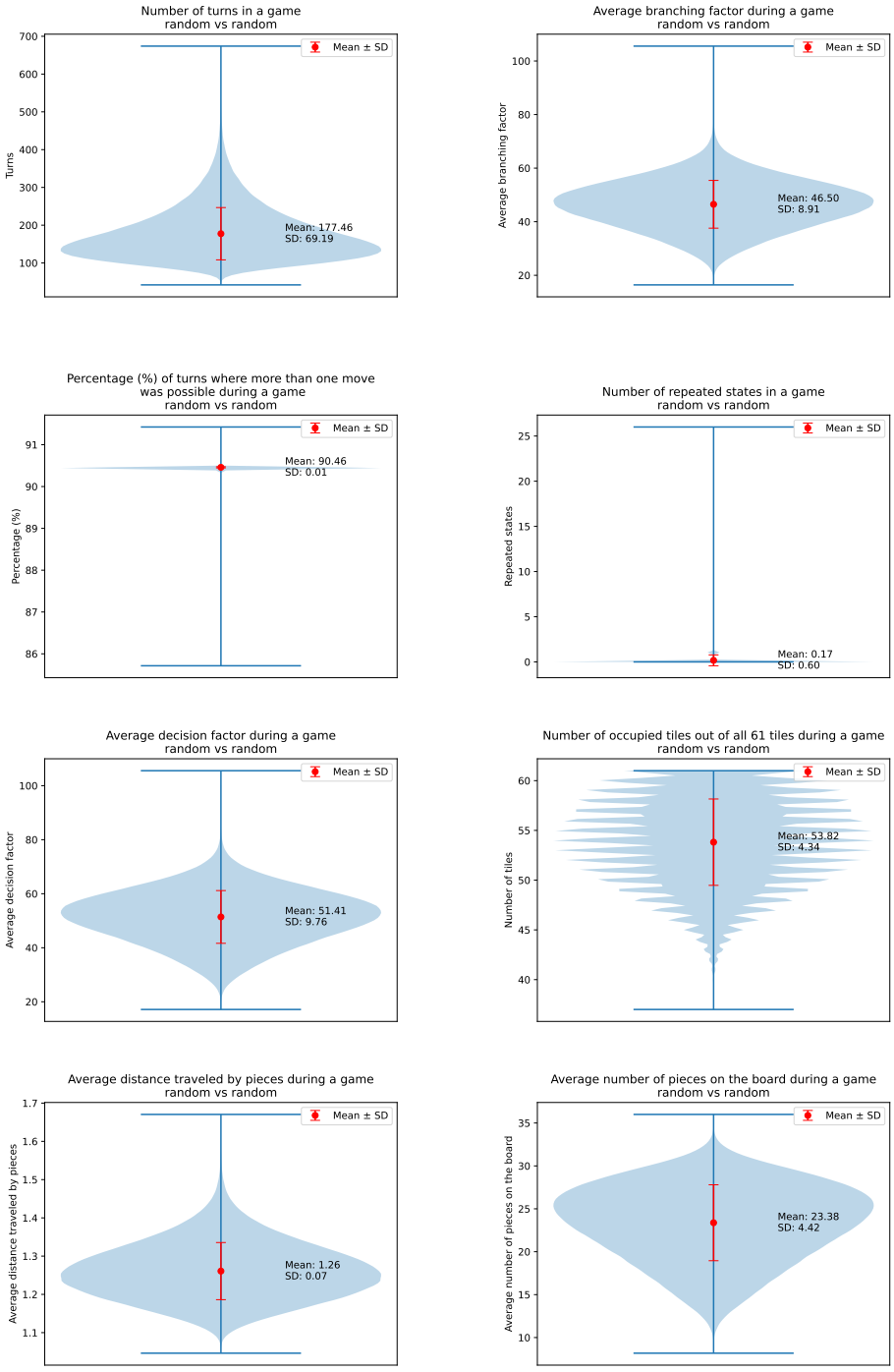
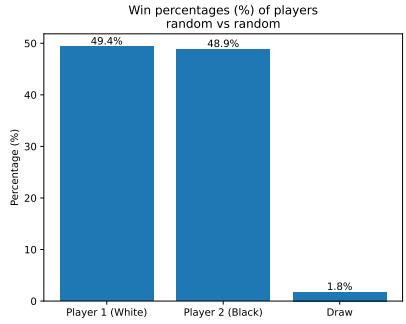
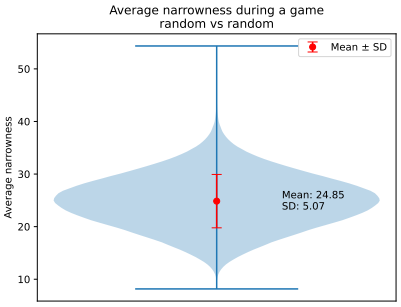
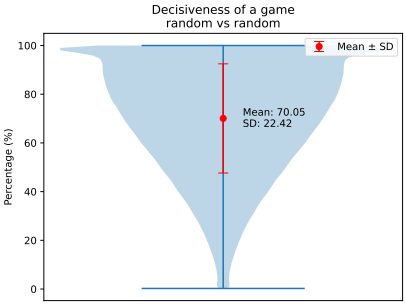

Number of turns in a game
random vs random
Turns
Mean ± SD
Mean: 177.46
SD: 69.19
Average branching factor during a game
random vs random
Average branching factor
Mean ± SD
Mean: 46.50
SD: 8.91
Percentage (%) of turns where more than one move
was possible during a game
random vs random
Percentage (%)
Mean ± SD
Mean: 90.46
SD: 0.01
Number of repeated states in a game
random vs random
Repeated states
Mean ± SD
Mean: 0.17
SD: 0.60
Average decision factor during a game
random vs random
Average decision factor
Mean ± SD
Mean: 51.41
SD: 9.76
Number of occupied tiles out of all 61 tiles during a game
random vs random
Number of tiles
Mean ± SD
Mean: 53.82
SD: 4.34
Average distance traveled by pieces during a game
random vs random
Average distance traveled by pieces
Mean ± SD
Mean: 1.26
SD: 0.07
Average number of pieces on the board during a game
random vs random
Average number of pieces on the board
Mean ± SD
Mean: 23.38
SD: 4.42

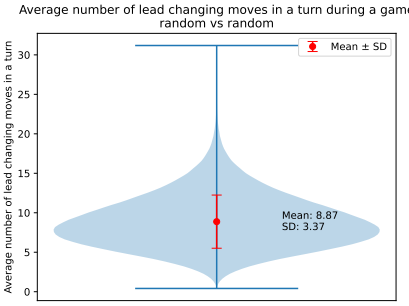
Figure A.1

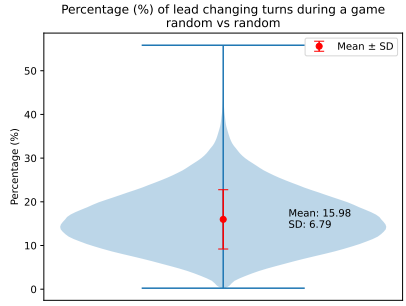
A

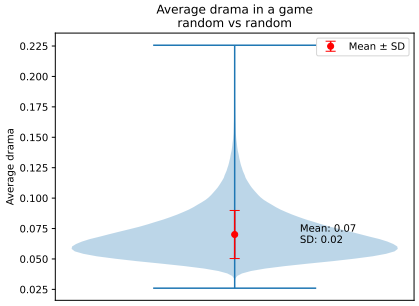

Win percentages (%) of players
random vs random
49.4%
48.9%
1.8%
Percentage (%)
Player 1 (White)
Player 2 (Black)
Draw

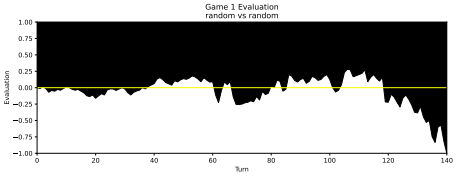

Average narrowness during a game
random vs random
Mean ± SD
Mean: 24.85
SD: 5.07
Average narrowness


Decisiveness of a game
random vs random
Mean ± SD
Mean: 70.05
SD: 22.42
Percentage (%)


Average number of lead changing moves in a turn during a game
random vs random
Mean ± SD
Mean: 8.87
SD: 3.37
Average number of lead changing moves in a turn


Percentage (%) of lead changing turns during a game
random vs random
Mean ± SD
Mean: 15.98
SD: 6.79
Percentage (%)


Average drama in a game
random vs random
Mean ± SD
Mean: 0.07
SD: 0.02
Average drama


Game 1 Evaluation
random vs random
Evaluation
Turn

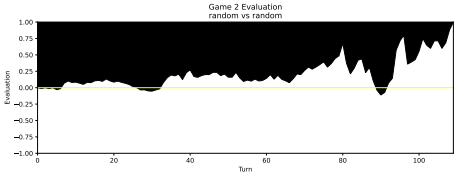
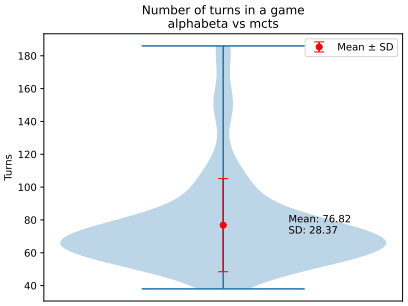
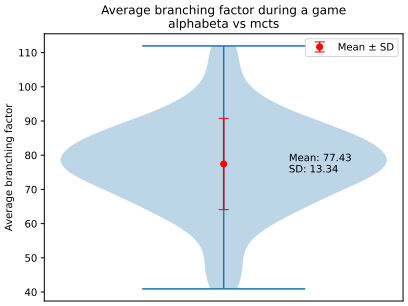

Game 2 Evaluation
random vs random
Evaluation
Turn

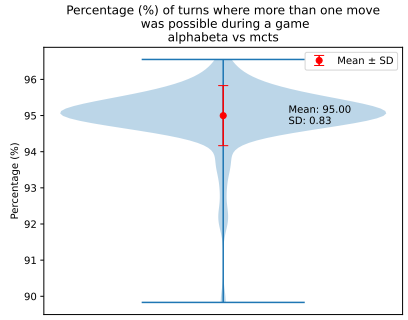
Figure A.2

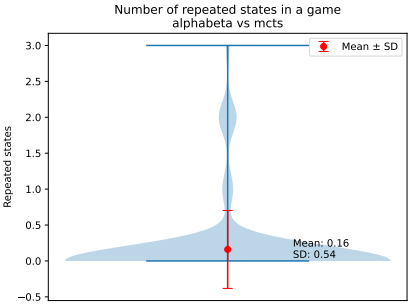
A.2 Alpha-Beta Agent vs MCTS Agent simulations

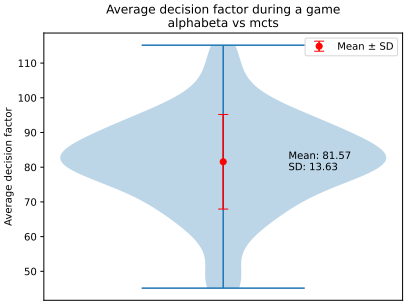
A

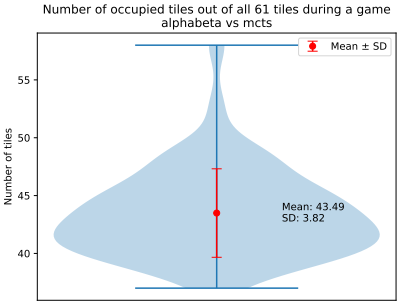

Number of turns in a game
alphabeta vs mcts
Mean ± SD
Turns
Mean: 76.82
SD: 28.37

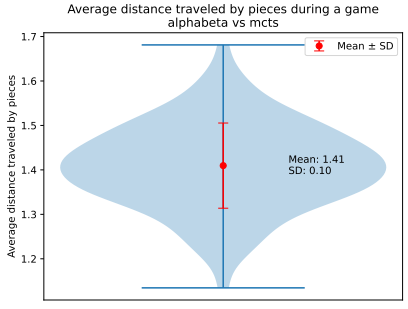

Average branching factor during a game
alphabeta vs mcts
Mean ± SD
Average branching factor
Mean: 77.43
SD: 13.34


Percentage (%) of turns where more than one move
was possible during a game
alphabeta vs mcts
Mean ± SD
Percentage (%)
Mean: 95.00
SD: 0.83


Number of repeated states in a game
alphabeta vs mcts
Mean ± SD
Repeated states
Mean: 0.16
SD: 0.54


Average decision factor during a game
alphabeta vs mcts
Mean ± SD
Average decision factor
Mean: 81.57
SD: 13.63


Number of occupied tiles out of all 61 tiles during a game
alphabeta vs mcts
Mean ± SD
Number of tiles
Mean: 43.49
SD: 3.82


Average distance traveled by pieces during a game
alphabeta vs mcts
Mean ± SD
Average distance traveled by pieces
Mean: 1.41
SD: 0.10

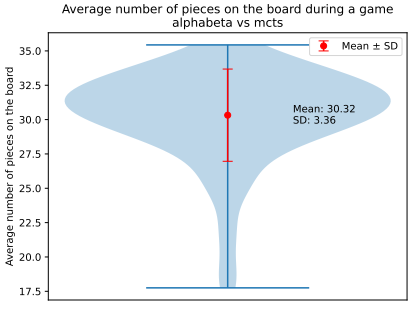

Average number of pieces on the board during a game
alphabeta vs mcts
Mean ± SD
Average number of pieces on the board
Mean: 30.32
SD: 3.36

Figure A.3

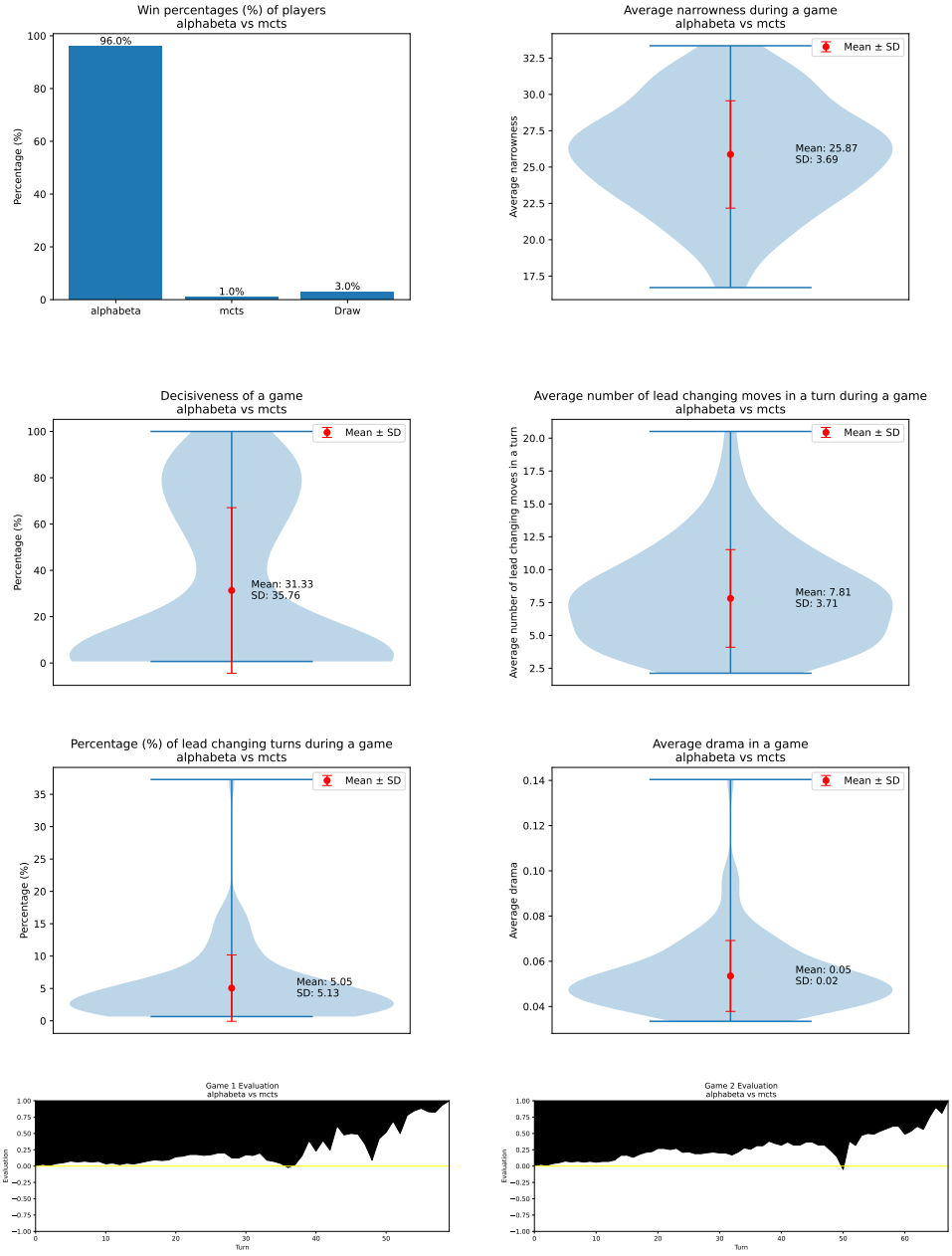

Win percentages (%) of players
alphabeta vs mcts
Percentage (%)
96.0%
1.0%
3.0%
alphabeta
mcts
Draw
Average narrowness during a game
alphabeta vs mcts
Average narrowness
Mean ± SD
Mean: 25.87
SD: 3.69
Decisiveness of a game
alphabeta vs mcts
Percentage (%)
Mean ± SD
Mean: 31.33
SD: 35.76
Average number of lead changing moves in a turn during a game
alphabeta vs mcts
Average number of lead changing moves in a turn
Mean ± SD
Mean: 7.81
SD: 3.71
Percentage (%) of lead changing turns during a game
alphabeta vs mcts
Percentage (%)
Mean ± SD
Mean: 5.05
SD: 5.13
Average drama in a game
alphabeta vs mcts
Average drama
Mean ± SD
Mean: 0.05
SD: 0.02
Game 1 Evaluation
alphabeta vs mcts
Evaluation
Turn
Game 2 Evaluation
alphabeta vs mcts
Evaluation
Turn

Figure A.4

Bibliography

References

[1] Claude E. Shannon. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950.

[2] Cameron Browne and Frederic Maire. Evolutionary game design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.

[3] Éric Piette, Dennis J.N.J. Soemers, Matthew Stephenson, Chiara F. Sironi, Mark H.M. Winands, and Cameron Browne. Ludii - the ludemic general game system. In *2020 European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 411–418. IOS Press, 2020.

[4] Eric Piette, Walter Crist, Dennis J. N. J. Soemers, Lisa Rougetet, Summer Courts, Tim Penn, and Achille Morenville. Gametable cost action: kickoff report. *ICGA Journal*, 46:11–27, 2024. doi:10.3233/ICG-240245.

[5] Walter Crist, Eric Piette, Karen Jeneson, Dennis J. N. J. Soemers, Matthew Stephenson, Luk van Goor, and Cameron Browne. Ludus coriovalli: using artificial intelligence-driven simulations to identify rules for an ancient board game. *Antiquity*, 2026. doi:10.15184/aqy.2025.10264.

[6] Lisa Rougetet and Tiago Hirth. Maurice kraïtchik (1882–1957) and his contributions to recreational mathematics. *SSRN Electronic Journal*, 2025. doi:10.2139/ssrn.6074494.

[7] Maurice Kraitchik. Tricolor. Belgian patent No. 372711, 1930. Royaume de Belgique.

[8] Maurice Kraitchik. *La mathématique des jeux, ou Récréations mathématiques*, volume 3. Imprimerie Stevens frères, 1930.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 3 edition, 2010.

[10] Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[11] Cameron Browne, Eric Piette, Walter Crist, Matthew Stephenson, and Dennis J. N. J. Soemers. Report on the 2nd digital ludeme project workshop. *ICGA Journal*, 2022. URL: https://journals.sagepub.com/doi/10.3233/ICG-220211, doi:10.3233/ICG-220211.

[12] Ryan B. Hayward and Jack Van Rijswijck. Hex and combinatorics. *Discrete Mathematics*, 306(19–20):2515–2528, 2006.

[13] Tiago Hirth and Lisa Rougetet. Mathematical games from the sphinx 1931–1939. Unpublished paper. Supported by COST Action CA22145 – GameTable | Computational Techniques for Tabletop Games Heritage. URL: https://gametable.network/.

[14] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[15] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[16] Éric Piette, Dennis J. N. J. Soemers, Matthew Stephenson, Chiara F. Sironi, Mark H. M. Winands, and Cameron Browne. Ludii — the ludemic general game system. In *Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 411–418. IOS Press, 2020. doi:10.3233/FAIA200120.

[17] Walter Crist, Matthew Stephenson, Éric Piette, and Cameron Browne. The ludii games database: A resource for computational and cultural research on traditional board games. *DHQ: Digital Humanities Quarterly*, 18(4), 2024.

[18] Éric Piette, Lisa Rougetet, Walter Crist, Matthew Stephenson, Dennis J. N. J. Soemers, and Cameron Browne. A ludii analysis of the french military game. *Board Game Studies*, 2021.

[19] Walter Crist, Éric Piette, Dennis J. N. J. Soemers, Matthew Stephenson, and Cameron Browne. Computational approaches for recognising and reconstructing ancient games: The case of ludus latrunculorum. In Véronique Dasen and Marco Vespa, editors, *The Archaeology of Play: Material Approaches to Games and Gaming in the Ancient World*. Oxbow, Oxford, 2024.

[20] Walter Crist and Éric Piette. Changing the game: Measuring mesopotamian games with ai-simulated play. In *Proceedings of the European Association of Archaeologists Annual Meeting*, 2025.

[21] Walter Crist, Éric Piette, Karen Jeneson, Dennis J. N. J. Soemers, Matthew Stephenson, Luk van Goor, and Cameron Browne. Ludus coriovalli: using artificial intelligence-driven simulations to identify rules for an ancient board game. *Antiquity*, 2026. doi:10.15184/aqy.2025.10264.

[22] Dorina Moullou, Walter Crist, Timothy Penn, and Éric Piette. Gametable network: Unveiling the past, embracing the future through ai-driven archaeological research. In *Proceedings of the 30th Annual Meeting of the European Association of Archaeologists*, 2024.

[23] Dennis J. N. J. Soemers, Jakub Kowalski, Walter Crist, Summer Courts, Tim Penn, and Eric Piette. Bridging ai and cultural heritage: Outcomes from the gametable wg1 london meeting. *ICGA Journal*, 47(1):41–47, 2025. doi:10.1177/13896911251329543.

[24] Cameron Bolitho Browne. *Automatic Generation and Evaluation of Recombination Games*. PhD thesis, Queensland University of Technology, 2008.

[25] Matthew Stephenson, Dennis J. N. J. Soemers, Eric Piette, and Cameron Browne. General game heuristic prediction based on ludeme descriptions. In *IEEE Conference on Games (CoG)*, 2021. URL: https://ieeexplore.ieee.org/document/9619052.

[26] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

[27] Dennis J. N. J. Soemers. Learning state-action features for general game playing, 2023.

[28] Ludii Portal, Concepts https://ludii.games/searchConcepts.php.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**