UCLouvain

École polytechnique de Louvain

# Game-Specific vs. Generalized MCTS Approaches

## A Study Using Ludii's Game Library

Author: **Gauwain Savary–Kerneïs**
Supervisor: **Éric Piette**
Readers: **Hélène Verhaeghe**, **Quentin Cappart**, **Achille Morenville**
Academic year 2025–2026
Master [120] in Computer Science and Engineering

# Contents

# Summary

This thesis investigated whether measurable game properties can be used to predict effective Monte Carlo Tree Search (MCTS) configurations in General Game Playing, and whether such predictions can be exploited to improve Best Agent Identification procedures.

To support this study, a large-scale experimental framework was developed within the Ludii General Game System, enabling systematic evaluation of MCTS variants across hundreds of structurally diverse games under controlled experimental conditions. Both One-Factor-at-a-Time and multi-component combination experiments were conducted, producing datasets for supervised prediction and warm-start transfer analyses.

The results show that game-property-based prediction captures meaningful coarse performance trends across games. Tree-based models consistently outperform linear baselines, indicating that the relationship between game structure and MCTS performance is at least partially nonlinear. However, predictive accuracy remains insufficient for reliable direct best-agent recommendation. Multiple analyses suggest that the primary limitation lies not only in data quantity, but also in the expressiveness of the available game representations. Learning curves plateau at larger training sizes, simple feature engineering provides little improvement, and PCA and clustering analyses reveal only coarse separability with substantial overlap in feature space.

These findings reinforce the broader challenge of adaptive search in General Game Playing: selecting effective search behavior from limited prior knowledge about a game remains difficult even when large-scale structural descriptors are available.

Despite these limitations, the experiments also show that predictive models remain practically useful when used as weak priors rather than as final decision mechanisms. Warm-start Best Agent Identification experiments demonstrate that fractional prior initialization can improve early-budget search efficiency while still allowing rapid online correction when predictions are inaccurate. This suggests that offline learning and online adaptation should be viewed as complementary components rather than competing alternatives.

Overall, a main contribution of this work is methodological. Rather than relying on direct zero-shot prediction of the best MCTS configuration, the results support a hybrid approach in which predictive models provide low-commitment guidance while adaptive online search remains responsible for robust final selection.

More broadly, the results suggest that future progress in adaptive MCTS configuration may depend less on increasingly complex predictive models than on richer representations of game dynamics and tighter integration between offline learning and online adaptation.

# Acknowledgments

I would like to express my sincere gratitude to my supervisor, Éric Piette, for his guidance, availability, and insightful feedback throughout this project. His expertise in the Ludii platform and General Game Playing was invaluable in shaping both the research questions and the experimental framework developed in this thesis.

I also thank the readers, Hélène Verhaeghe, Quentin Cappart, and Achille Morenville, for their time and thoughtful evaluation of this work.

Finally, I am grateful to the CECI computing infrastructure for providing the computational resources that made the large-scale experiments presented here possible.

# 1

# Introduction

## 1.1 Context and Motivation

Monte Carlo Tree Search (MCTS) [1, 2] has become one of the most influential approaches for sequential decision-making in complex environments with large search spaces. Rather than relying on handcrafted evaluation functions or exhaustive enumeration, MCTS estimates the value of actions through randomized simulations combined with incremental tree expansion, balancing exploration and exploitation during search. Its flexibility and domain independence have led to successful applications across a wide variety of problems, including board games, real-time strategy games [3], combinatorial optimization [4], and planning tasks [5]. In particular, MCTS has achieved remarkable success in game-playing systems, most famously as a core component of AlphaGo [6]. Another important property of MCTS is its anytime behavior: the algorithm can return a valid decision at any point during computation, while solution quality generally improves as additional simulation budget becomes available.

In the context of General Game Playing (GGP) [7], where agents must play previously unseen games from formal rule descriptions alone, these properties make MCTS especially attractive. Because it requires little domain-specific knowledge beyond legal move generation and terminal-state evaluation, MCTS has become a leading approach for decision-making in modern GGP systems [8].

Despite its success, in the context of games, MCTS performance varies significantly depending on their characteristics. Properties such as board size, board topology, branching factor, average game length, and degree of tactical versus strategic decision-making all influence which algorithmic enhancements prove beneficial. To address these variations, researchers have proposed numerous MCTS variants targeting different phases of the algorithm. Each of these variants exhibits strengths and weaknesses that depend on the characteristics of the game being played. However, it remains unclear to what extent MCTS enhancements generalize across different games, and when game-specific adaptation becomes necessary. A major limitation of existing studies is that MCTS variants are often evaluated on relatively small or heterogeneous sets of games, frequently under differing experimental conditions and with limited coverage of variant combinations. As a result, it remains difficult to draw broad conclusions about which techniques are consistently

beneficial, how robust their gains are across game families, and how different enhancements interact when combined.

## 1.2 The Ludii Platform

The Ludii General Game System [9] provides a particularly well-suited platform for studying these questions. With a library of more than 1,400 games spanning diverse rulesets, complexity levels, board topologies, and strategic structures [10], Ludii enables systematic large-scale evaluations under controlled experimental conditions. Its Game Description Language [11] represents games as compositions of ludemes, modular game components that provide a uniform and extensible representation [12] across a highly diverse game corpus. In addition to game execution, Ludii exposes many structural and gameplay-related properties through its API [13], making automated feature extraction possible for every game in the library.

This combination of scale, diversity, standardized game representations, and programmatic access to game characteristics makes Ludii an especially valuable framework for investigating the relationship between game properties and MCTS performance.

## 1.3 Research Questions

The central problem addressed by this thesis is: **Can game properties help predict which Monte Carlo Tree Search variants are likely to perform well for a given game?**

To address this overarching problem, this thesis investigates the following specific research questions:

**RQ1. Variant effectiveness.** Which individual MCTS enhancements provide robust performance improvements across a diverse corpus of games, and which appear strongly game-dependent?

**RQ2. Combination effects.** How do different MCTS enhancements interact when combined? Are their effects complementary, redundant, or antagonistic?

**RQ3. Game property predictors.** Which measurable game properties, such as branching factor, average game length, board topology, or degree of tactical versus strategic play, are most predictive of MCTS variant performance?

**RQ4. Adaptive configuration.** Can predictions derived from game properties be used as useful priors to accelerate best-agent identification on previously unseen games, particularly under limited search budgets?

## 1.4 Contributions

This thesis makes the following contributions:

1. **A modular experimental framework** for systematic MCTS variant evaluation within the Ludii platform, supporting plug-and-play composition of selection, simulation, backpropagation, and final-move strategies. The framework is designed for large-scale parallel execution on computing clusters.

2. **A broad empirical study** comparing combinations of MCTS variants across a corpus of two-player, deterministic, alternating-move Ludii games.

3. **A game-property-based prediction analysis** using regression and multi-output regression models, including feature engineering, clustering, and PCA analyses to investigate the predictive power and limitations of the considered game feature representations for MCTS variant performance prediction.

4. **A warm-start transfer study** evaluating whether imperfect but informative predictions can be used as low-commitment priors for Best Agent Identification.

## 1.5 Thesis Organization

The remainder of this thesis is organized as follows.

**Chapter 2** introduces the background necessary for this work, including General Game Playing, the Ludii platform, Monte Carlo Tree Search, and prior work on MCTS variants and algorithm selection.

**Chapter 3** describes the experimental framework, including the modular MCTS implementation, game corpus selection, experimental design, and execution pipeline used throughout the study.

**Chapter 4** presents the empirical results, covering large-scale variant evaluation, game-property-based prediction experiments, feature-space analyses, and warm-start transfer experiments.

**Chapter 5** discusses the main findings, limitations, and implications of the results, and outlines directions for future work.

Finally,

**Chapter 6** concludes the thesis and summarizes its main contributions.

# 2 Background

This chapter introduces the theoretical and practical foundations underlying this thesis. We first present the General Game Playing setting and the Ludii platform [9], which together provide the experimental framework for this work. We then review the foundations of MCTS and the main families of algorithmic enhancements considered in this study. Next, we discuss how measurable game properties may influence search performance and motivate the use of game descriptors for prediction tasks. Finally, we review the prior work most closely related to this thesis, including large-scale MCTS evaluation in Ludii [14], heuristic performance prediction [15], and Best Agent Identification methods [16].

## 2.1 General Game Playing

General Game Playing (GGP) [7] is a subfield of Game AI concerned with building agents that can play arbitrary games competently, given only a formal description of the rules at runtime. Unlike game-specific AI systems, such as the chess engine Stockfish [17] or the Go-playing AlphaZero [18], a GGP agent receives no hand-crafted evaluation function and no domain-specific knowledge. It must reason about the game purely from its formal description and the experience it accumulates during play.

The challenge of GGP is both a planning challenge and a representation challenge. The agent must parse and interpret an unknown ruleset, identify legal actions, estimate the value of game states and select moves, all within strict time constraints. This demands algorithms that are both domain-independent and computationally tractable, two properties that are often in tension.

Early GGP research was driven by Stanford's Game Description Language (GDL) [7], which enabled structured competitions between 2005 and 2016/17 that were instrumental in advancing MCTS-based GGP agents [19, 20]. The Ludii General Game System [9], described in detail in Section 2.2, takes a different and more expressive approach: it encodes games using a purpose-built language grounded in the concept of *ludemes*, enabling both execution and analysis of over 1,400 games from a uniform representation, a breadth that was difficult to achieve under earlier GDL-based systems.

## 2.2 The Ludii General Game System

Ludii [9] is a complete general game system initially developed as part of the ERC-funded Digital Ludeme Project (DLP) [21], and whose development continues within the COST Action GameTable network (CA22145) [22]. Its central design principle is the *ludemic approach*: games are described as compositions of *ludemes*, modular conceptual units of game design that correspond to recognizable game features such as boards, pieces, movement rules, and ending conditions [12]. This approach simultaneously enables formal game specification, automated game analysis, and computational game-playing within a single unified framework [23].



### 2.2.1 The Game Description Language

Games in Ludii are encoded in a domain-specific Game Description Language (GDL) using a Lisp-like syntax. A Ludii game description consists of four main components: the `(game ...)` root ludeme identifying the game; `(equipment ...)` specifying the board geometry and pieces; `(rules ...)` defining the play, starting, and ending conditions; and optional `(metadata ...)` providing descriptive information, graphical settings, and AI hints.

Listing 2.1 shows an excerpt of the official Ludii description of Breakthrough, a two-player abstract strategy game in which each player controls a row of pawns that advance toward the opponent's side, winning by reaching the back rank. The source is the Ludii repository game file.[1]

Example 2.1: Ludii description of Breakthrough.

```
(game "Breakthrough"

    (players {(player N) (player S)})

    (equipment
        {

        (board (square 8))

        (piece "Pawn"
            Each
            (or
                {
                (move Step Forward (to if:(is Empty (to))))
                (move
                    Step
                    (directions { FR FL })
                    (to
                        if:(or
                            (is Empty (to))
                            (is Enemy (who at:(to)))
                        )
                        (apply (remove (to)))
                    )
                )
                }
            )
        )
    )

```

[1] https://github.com/Ludeme/Ludii/blob/master/Common/res/lud/board/race/reach/Breakthrough.lud

```
    (regions P1 (sites Top))
    (regions P2 (sites Bottom))

    }
)

(rules
    (start
        {
            (place "Pawn1" (expand (sites Bottom)))
            (place "Pawn2" (expand (sites Top)))
        }
    )

    (play (forEach Piece))

    (end (if (is In (last To) (sites Mover)) (result Mover Win)))
)

)
```

Board geometries in Ludii are first-class objects: the `(board ...)` ludeme supports square, hexagonal, triangular, and many other topologies [24], as well as irregular graph-based layouts for historical games. This structural diversity is one of the properties that makes Ludii uniquely interesting as an experimental platform for game AI.

### 2.2.2 The Game Library

As of the time of writing, Ludii's library contains over 1,400 games[2] spanning thousands of years of human game-playing history, from ancient Egyptian race games to modern abstract strategy games [10]. The library includes games from every inhabited continent and covers a wide range of strategic and mechanical complexity. Ludii also serves as a competition platform for GGP research [25], enabling structured agent benchmarking across diverse game sets. Indeed, Ludii has already been applied to the study of historical and ancient games using AI techniques, including MCTS-based approaches [26–32], illustrating that the challenge of selecting effective MCTS configurations extends even to historical game reconstruction contexts. This diversity is precisely what makes Ludii valuable for our study: a meaningful comparison of MCTS variants requires a corpus broad enough to expose the conditions under which each variant thrives or fails.

For this thesis, we restrict our attention to two-player, deterministic, perfect-information, alternating-move games, a subset that still encompasses 77% of games in the Ludii library while maintaining a controlled evaluation setting in which win rate is a well-defined and unambiguous metric.

### 2.2.3 Programmatic Access to Game Properties

A key feature of Ludii for this work is its Java API[3], which exposes game properties programmatically [33]. Given a loaded `Game` object, one can query structural features such as board size, number of graph elements, whether the game uses stacking, whether it involves captures, whether it has a line-based winning condition, and many others [13].

[2] When including all rulesets obtainable through Ludii's option and ruleset system, the total number of game variants exceeds two million.

[3] `https://github.com/Ludeme/Ludii`

This enables automated feature extraction for every game in the corpus, forming the basis of the game property analyses described in Chapter 4.

## 2.3 Monte Carlo Tree Search: Foundations



MCTS [1, 2] is a family of best-first tree search algorithms that estimate the value of game states through playouts rather than analytical evaluation. In the context of General Game Playing (GGP) [7], where agents must play previously unseen games from formal rule descriptions alone, MCTS has become a leading approach for decision-making in modern GGP systems [34], thanks to its domain independence and ability to scale with available computation.

### 2.3.1 The Four Phases

At its core, MCTS faces the classic exploration-exploitation dilemma: it must balance exploiting nodes that have historically yielded good results with exploring less-visited regions of the search tree that may prove better.

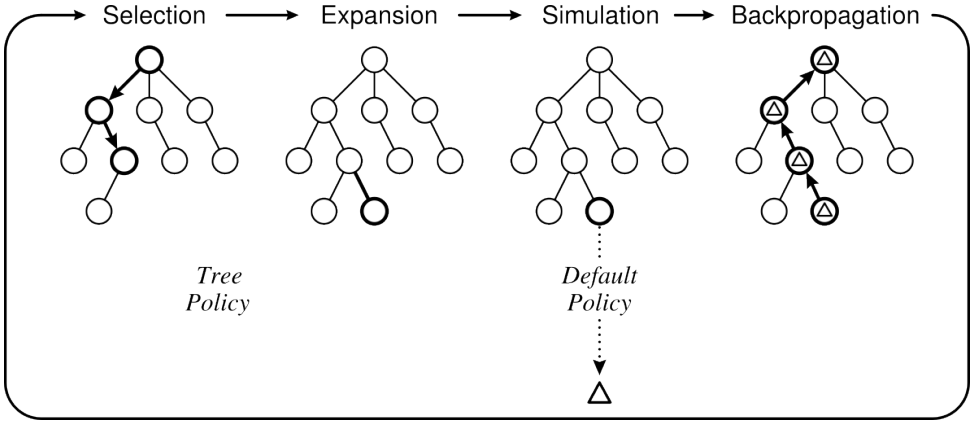


Figure 2.1: The four phases of Monte Carlo Tree Search. Reproduced from [1].

A standard MCTS iteration consists of four sequential phases:

**Selection.** Starting from the root node (current game state), the algorithm traverses the existing tree by repeatedly applying a *tree policy* that balances exploitation of high-value nodes with exploration of less-visited ones. This continues until a node is reached that is either terminal or not fully expanded.

**Expansion.** One or more previously unexplored child nodes are then added to the search tree. In the standard formulation, a single child is added per iteration; some variants expand all children at once.

**Simulation.** From the newly expanded node, a *default policy* (commonly random uniform action selection) plays out the game to a terminal state. This playout produces a game result (win, loss, or draw).

**Backpropagation.** The result of the simulation is propagated back up the tree, updating visit counts and value estimates for every node on the path from the expanded node to the root.

These four phases are repeated for as many iterations as the available time or computation budget permits. Over successive iterations, the search tree progressively concentrates on the most promising regions of the search space while continuing to explore less-visited alternatives. Once the budget is exhausted, a *final move selection* strategy determines which action is returned from the root node.



### 2.3.2 The UCT Algorithm

The most widely used tree policy is UCT (Upper Confidence bounds applied to Trees), introduced by Kocsis and Szepesvári [35] by adapting the UCB1 bandit algorithm [36] to tree search. The UCT selection criterion applied at each internal node selects the child $i$ maximizing:

$$\mathrm{UCT}(i) = \frac{w_i}{n_i} + C\sqrt{\frac{\ln N}{n_i}} \tag{2.1}$$

where $w_i$ is the total score accumulated from simulations through child $i$, $n_i$ is the number of visits to child $i$, $N$ is the number of visits to the current node, and $C > 0$ is an exploration constant controlling the exploration-exploitation trade-off. The first term is the exploitation component (average value of child $i$), and the second is the exploration bonus (favoring less-visited children).

UCT inherits the exploration-exploitation principles of UCB1 and provides asymptotic convergence guarantees under appropriate conditions [35]. In practice, however, its performance is highly sensitive to the choice of the exploration constant $C$, which often requires empirical tuning for different domains.

### 2.3.3 Properties of MCTS

Several properties make MCTS particularly well suited to General Game Playing.

**Domain independence.** Standard MCTS requires only the ability to enumerate legal actions, apply actions to game states, and detect terminal states. Unlike traditional minimax-based approaches, it does not rely on handcrafted evaluation functions, making it applicable to a wide range of games without domain-specific adaptation.

**Anytime behavior.** MCTS can return a valid decision at any point during computation, with decision quality generally improving as additional simulations are performed. This property is especially important in time-constrained settings.

**Selective tree growth.** Unlike fixed-depth search methods, MCTS allocates more simulations to promising regions of the search space while still exploring alternative actions. This allows computational effort to be distributed adaptively rather than uniformly across the tree.

**Scalability.** Because MCTS is simulation-based, it naturally benefits from increased computational budget and can be parallelized effectively.

These advantages also come with limitations. In particular, MCTS may struggle in games with deep tactical sequences, very long horizons, or extremely large branching

factors, where playouts provide weak or noisy estimates of state quality. Many of the MCTS variants discussed in this thesis were proposed specifically to address these limitations.

## 2.4 Game Properties and Algorithm Performance



A central hypothesis of this thesis is that the relative performance of MCTS variants is not arbitrary but is systematically related to measurable properties of the game being played. This idea connects to two well-established theoretical concepts. The *algorithm selection problem*, formalized by Rice [37], asks which algorithm from a portfolio should be applied to a given problem instance based on measurable instance features. The *No Free Lunch theorem* [38] formalizes the impossibility of a universally superior algorithm over all possible problem distributions, providing theoretical grounding for why adaptive, property-driven selection is necessary in practice.

### 2.4.1 Game Features in Ludii

Ludii exposes a wide range of measurable game characteristics that can be used as feature representations. The work of Piette et al. [13] introduces a broad set of game concepts describing games along multiple dimensions, ranging from high-level structural properties to specific gameplay mechanics. These concepts can be divided into two broad categories.

*Static features* are derived directly from the game description without executing games. They include structural properties such as board size, board topology, number of piece types, and the presence of mechanics such as captures, promotions, or line-based winning conditions.

*Playout concepts* are estimated empirically by running games and collecting statistics from gameplay traces. Examples include average game length, average branching factor, frequencies of specific move types, and draw rates. In the work of Soemers et al. [14], these quantities were estimated using games played between random agents, providing a fast and domain-independent approximation.

Together, these features provide a rich characterization of games in the Ludii library and form the basis of the predictive analyses explored in this thesis.

### 2.4.2 Theoretically Motivated Predictors

Several measurable game characteristics are theoretically expected to influence the effectiveness of different MCTS enhancements.

**Branching factor.** The average number of legal moves per position. High branching factors make deep tree exploration expensive and reduce the frequency with which any individual action is revisited, weakening AMAF-based statistics (RAVE, GRAVE), which rely on actions being encountered frequently. Progressive Widening is specifically designed to address high branching factors by throttling expansion.

**Game length.** The average number of moves in a complete game. In long games, random playouts are unlikely to produce terminal states that reflect the strategic value of the opening position, degrading the playout signal. Simulation enhancements such as MAST and NST are theoretically most valuable in this regime, since they bias playouts toward historically profitable moves rather than selecting uniformly at random.

**Board topology.** The graph structure of the board affects move connectivity and the degree to which the AMAF assumption holds. In highly connected topologies, the same move may be legal from many positions, increasing the informativeness of global action statistics. In linear or sparse topologies, moves are more position-specific and AMAF generalisation is less warranted.

**Tactical vs. strategic character.** Games with strong short-horizon tactics, where a single oversight leads to immediate material loss, differ from games where strategic advantages accumulate gradually. Backpropagation enhancements like Implicit Minimax, which incorporate depth-limited minimax estimates, are theoretically most beneficial in tactically sharp positions where random playouts frequently miss critical threats.

**Win condition type.** Whether the game is decided by alignment (line-based), capture (elimination-based), territory (area-based), or scoring. Win condition type affects how reliably a random playout outcome correlates with the positional quality of the node from which it was launched.

**First-player advantage.** Soemers et al. [14] found that the estimated advantage of the first player in random play is a surprisingly reliable predictor of outcomes between MCTS agents, suggesting that the structural biases of a game toward one player tend to persist regardless of the strength of the search algorithm used.

## 2.5 Prior Work

Two lines of prior work are particularly relevant to this thesis. The first concerns large-scale empirical studies of MCTS variant performance in Ludii and the use of game properties for performance prediction. The second concerns Best Agent Identification methods designed to efficiently identify strong agent configurations under limited evaluation budgets.

### 2.5.1 MCTS Variant Performance in Ludii — Soemers et al. (2024)

Soemers et al. [14] conducted a large-scale empirical study characterising MCTS variant performance across the Ludii game library. They evaluated 61 MCTS agent configurations, constructed by crossing four selection strategies with three exploration constants and five playout strategies, were evaluated across 1,494 Ludii games, though computational constraints meant only approximately 10% of all scheduled matchups were completed.

A Random Forest trained on game features and one-hot agent encodings outperforms a dummy baseline, though with substantial cross-game variance. SHAP analysis reveals two robust findings: first-player advantage from random play is the single most predictive feature of MCTS outcomes, suggesting structural game biases persist regardless of the algorithm used; and very short truncated playouts (zero or four moves, without a value function) are reliably predictive of poor performance across all games.

The main limitations are sparse data coverage, an agent design space restricted to four selection and five playout strategies with no backpropagation variants, and feature analysis that examines predictors individually rather than in combination. This thesis extends this line of work by expanding the variant space to include backpropagation strategies and a wider selection of simulation policies, running a controlled evaluation across a diverse game corpus, and going beyond individual feature analysis to examine combination effects, learning curves, and clustering structure.

### 2.5.2 Best Agent Identification — Stephenson et al. (2025)

Stephenson et al. [16] address the problem of efficiently identifying the best-performing agent for each game in a large corpus under a limited trial budget. They formulate this as a multi-bandit best-arm identification setting [39] in which each game defines a bandit problem and each candidate agent corresponds to an arm, and propose the *Regret Change Potential* (RCP) algorithm to solve it.



RCP selects the next game-agent pair to evaluate by estimating, for each pair, how much a new trial could change the current simple regret, using a confidence interval to bound each arm's expected reward. Pairs where the ranking is still uncertain are prioritised; pairs whose outcome is already clear are deprioritised. RCP is *stop-anytime* and supports dynamic addition of new games or agents during search. It substantially outperforms prior BAI algorithms, including GapE, UCB-E, and Sequential Halving, on both the GVGAI and Ludii benchmarks.

Whereas the approach of Stephenson et al. relies purely on adaptive online evaluation, the present work investigates whether game-property-based predictions can provide informative priors for the search process. In this thesis, RCP is used as the online Best Agent Identification procedure in the warm-start experiments presented in Chapter 4, where the central question is whether offline predictions can improve early search efficiency while still allowing rapid correction when those predictions are inaccurate.

### 2.5.3 Heuristic Performance Prediction in Ludii — Stephenson et al. (2021)

Stephenson et al. [15] investigated a problem closely related to the one addressed in this thesis: rather than predicting which *MCTS variant* performs best, they studied whether the effectiveness of game-playing *heuristics* can be predicted directly from game descriptions within the Ludii system. Using ludemes as input features, several regression models were trained to estimate heuristic performance across a large corpus of games, enabling the selection of suitable heuristics without requiring exhaustive evaluation on each new game. Results demonstrated that ludeme-based representations carry useful predictive signal for this task.

This work is directly relevant to the present thesis in two ways. First, it establishes that Ludii's ludeme-based game representations can support supervised prediction of algorithmic performance, providing prior evidence that the feature extraction approach adopted here is a principled starting point. Second, it highlights a key limitation shared with our setting: even when coarse predictive trends are learnable, fine-grained discrimination between similarly performing configurations remains difficult. The present work extends this line of inquiry by targeting MCTS variant selection rather than heuristic selection, expanding the variant space to include selection, simulation, and backpropagation strategies, and explicitly investigating the structural reasons why predictive performance plateaus.

### 2.5.4 Algorithm Selection for Games

Prior work has also explored adaptive approaches for configuring search algorithms in General Game Playing. Sironi et al. [40] investigated online parameter tuning for MCTS, showing that important search parameters such as the UCT exploration constant can be highly game-dependent and can benefit from adaptation during play. This line of work

is complementary to the approach considered in this thesis, where adaptation is guided by offline predictions derived from game properties rather than exclusively from online search statistics.

More broadly, research on algorithm selection [37] and algorithm portfolios [41] has shown that selecting algorithms based on measurable problem characteristics can outperform any fixed algorithm across a diverse set of tasks. The present work applies this general perspective to the selection of MCTS configurations in General Game Playing environments.

Research on adversarial search pathologies and sampling-based planning [42, 43] further suggests that structural properties of search spaces can strongly influence the effectiveness of sampling-based search algorithms, providing additional motivation for studying the relationship between game characteristics and MCTS performance.

The following chapter describes the experimental framework built to conduct this study at scale.

# 3



# Experimental Framework

This chapter describes the experimental framework developed to evaluate MCTS variants at scale within the Ludii platform. The framework was designed with four main objectives: *modularity*, so that any combination of selection, simulation, backpropagation, and final-move strategy can be assembled without modifying the core engine; *reproducibility*, so that every experiment is fully specified through declarative configuration files and can be re-run identically; *scalability*, so that the workload can be distributed across a computing cluster with minimal manual coordination; and *auditability*, so that the coverage of the experiment plan and the properties of the game corpus can be inspected before any computation is spent running games.

Ludii provides a particularly suitable environment for this type of large-scale experimentation due to its unified game representation, broad game library, and efficient execution model. Previous work comparing Ludii with other General Game Playing systems has also demonstrated its suitability for large-scale empirical evaluation across diverse game sets [44].

The framework is implemented in Java 17 and interfaces directly with the Ludii engine through its public API [33]. The complete source code is publicly available at `https://github.com/Gauwal/MCTSGeneralVSGameSpecificThesis`.

## 3.1 Architecture Overview

Figure 3.1 illustrates the five-stage experiment pipeline. Each stage is implemented as an independent, single-responsibility command-line tool that reads from and writes to well-defined CSV files. This design means that any stage can be inspected, rerun, or replaced without affecting the others, and that all intermediate artefacts are human-readable.

**Stage 1: Game Catalog Build.** `BuildGameCatalog` enumerates all Ludii games, loads each one, extracts structural and topological properties, and writes a persistent `game_catalog.csv`. This stage is run once and its output is treated as a stable database for all subsequent stages.

**Stage 2: Test Plan Generation.** `GenerateTestPlan` reads the game catalog, constructs a set of planned experiments according to a configurable experimental design

(One-Factor-At-a-Time or Full Combination), and writes a `planned_tests.csv` specifying every matchup to be run.

**Stage 3: SLURM Script Generation.** `GenerateSlurmScripts` reads the test plan and the game catalog and emits one SLURM `sbatch` script per planned test, plus a master `submit_all.sh` script. Resource requirements (CPU count, memory, wall-clock time) are estimated deterministically from the game's catalog properties and the planned time budget.

**Stage 4: Experiment Execution.** `RunPlannedTest` executes a single planned test identified by its test ID: it loads the game, instantiates the baseline and variant agents, runs the specified number of games with alternating starting positions, and writes a result CSV row. On the cluster each SLURM job calls this tool for exactly one test.

**Stage 5: Data Analysis.** Python scripts parse and aggregate result CSVs, apply timeout-aware post-processing, train predictive models, and generate the figures and summary tables used in the Results chapter.

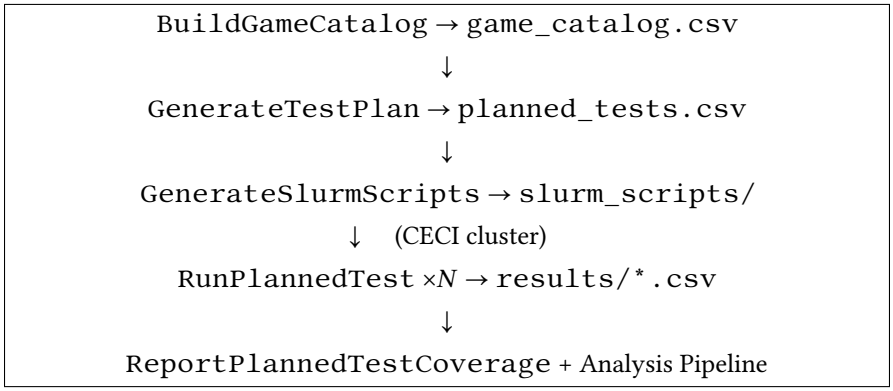


Figure 3.1: The five-stage experiment pipeline. Each stage communicates exclusively through CSV files, making every intermediate artefact inspectable and every stage independently reproducible.

## 3.2 MCTS Agent Implementation

All agents are implemented through a single class, `MCTSVariations`, built on top of Ludii's native MCTS framework. The class exposes a modular configuration interface in which each agent is defined by four independent components: a selection strategy, a simulation strategy, a backpropagation strategy, and a final move selection strategy. This design makes it possible to evaluate a large variety of MCTS configurations while maintaining a uniform implementation framework.

Listing 3.1 illustrates the instantiation of a complete MCTS configuration.

Example 3.1: Instantiating an MCTS configuration via `MCTSVariations`.

```
1 MCTSVariations agent = new MCTSVariations(
```

```
    "Progressive Widening",   // selection strategy
    "MAST",                   // simulation strategy
    "Score Bounded",          // backpropagation strategy
    "Robust"                  // final move selection
);
```

Strategy identifiers are matched case-insensitively and support multiple aliases to ensure robust parsing of experimental configurations. Unknown strategy identifiers are automatically mapped to the corresponding baseline strategy, and all such fallbacks are recorded in execution logs for auditing purposes.

Tree reuse between moves is disabled for all agents (`setTreeReuse(false)`), ensuring that each move decision is computed from a freshly initialized search tree. This avoids reusing search information across consecutive game states and simplifies interpretation of experimental results, at the cost of some playing strength relative to tree-reusing configurations.

The evaluated configurations span both classical MCTS enhancements and hybrid search methods integrated within Ludii's MCTS framework. *Ludii built-in* strategies are natively available through Ludii's API; *Custom* strategies were implemented from scratch as part of this thesis. Tables 3.1, 3.2, and 3.3 summarize the implemented selection, simulation, and backpropagation strategies respectively.

Table 3.1: Selection strategies implemented in the framework.

| **Strategy** | **Key Mechanism** | **Source** | **Ref** |
|---|---|---|---|
| UCB1 | Standard UCB confidence bound | Built-in | [35, 36] |
| UCB1-Tuned | Variance-aware confidence bound | Built-in | [35, 36] |
| UCB1-GRAVE | GRAVE AMAF statistics with UCB1 exploration | Built-in | [45] |
| Progressive History | Global action history bias | Built-in | [46] |
| McBRAVE | Monte Carlo BRAVE heuristic | Built-in | [47] |
| McGRAVE | Monte Carlo GRAVE heuristic | Built-in | [45] |
| Progressive Widening | Throttled child expansion ($k \cdot n^{\alpha}$) | Custom | [48] |
| Implicit Minimax | UCB + minimax-backed value blending | Custom | [49] |
| Alpha-Beta | Alpha-Beta pruning integrated in selection | Custom | [49] |
| AG0 | AlphaGo Zero policy-prior guidance | Built-in | [50] |
| Noisy AG0 | AG0 with Dirichlet exploration noise | Built-in | [50] |
| ExIt | Expert Iteration policy selection | Built-in | [51] |
| Progressive Bias | Heuristic bias decaying with visit count | Built-in | [52] |

Selection strategies include classical UCB-based approaches such as UCB1 and UCB1-Tuned, AMAF-based extensions such as GRAVE and McGRAVE, history-based methods such as Progressive History, as well as more advanced approaches including Progressive Widening, Implicit Minimax, Alpha-Beta-guided selection, and policy-guided methods derived from AlphaGo Zero and Expert Iteration.

Table 3.2: Simulation strategies implemented in the framework.

| Strategy | Key Mechanism | Source | Ref |
|---|---|---|---|
| Random Playout | Uniform random moves until terminal state | Built-in | [1] |
| Heuristic Playout | Stochastic playout biased by heuristic evaluations | Built-in | [1] |
| Heuristic Sampling Playout | Sampling from heuristic-weighted move probabilities | Built-in | [1] |
| MAST | Bias moves using global move-value statistics | Built-in | [20] |
| NST | Prefer moves from successful short move sequences | Built-in | [34] |
| HS Playout | History-sensitive playout policy | Built-in | [1] |
| Last Good Reply | Replay replies shown effective in prior simulations | Custom | [53] |

Simulation strategies range from standard random playouts to progressively more informed policies, including heuristic-guided playouts, MAST, NST, history-sensitive playouts, and Last Good Reply.

Table 3.3: Backpropagation strategies implemented in the framework.

| Strategy | Key Mechanism | Source | Ref |
|---|---|---|---|
| Monte Carlo Backup | Average of simulation returns | Built-in | [1] |
| Heuristic Backup | Heuristic evaluations shape the backup value | Built-in | [1] |
| AlphaGo-style Backup | Blend of simulation returns with value priors | Built-in | [6] |
| Qualitative Bonus | Heuristic-based qualitative bonuses added to returns | Built-in | [54] |
| Score-Bounded Backup | Limits influence of extreme scores via score bounds | Custom | [55] |
| Implicit Minimax Backup | Blends Monte Carlo returns with minimax estimates | Custom | [49] |

Backpropagation strategies include standard Monte Carlo backup, heuristic-based backup rules, AlphaGo-style value blending, qualitative bonus methods, and custom score-bounded or implicit minimax backup variants.

Unless stated otherwise, all experiments use Robust Child as the final move selection strategy in order to isolate the effects of the other MCTS components.

## 3.3 Game Catalog and Corpus Selection

### 3.3.1 Catalog Construction

To support large-scale analysis and prediction experiments, a structured catalog of Ludii games was constructed by automatically extracting a set of measurable game descriptors from every successfully loaded game in the library. The extraction process relies exclusively on static analysis of game descriptions and board structures, without executing gameplay simulations.

For each game, structural, topological, and gameplay-related properties were collected and stored in a standardized tabular format. These descriptors include information about board geometry, graph connectivity, playable sites, player count, and the presence of specific mechanics such as stochasticity, stacking, hidden information, or heuristic support. Table 3.4 summarizes the main categories of extracted features used throughout this thesis.



Table 3.4: Features extracted for each game in the catalog.

| Category | Feature | Description |
|---|---|---|
| Structural | `numPlayers` | Number of players |
| | `numCells` | Number of cells in the board graph |
| | `numVertices` | Number of vertices |
| | `numEdges` | Number of edges |
| | `numRows`, `numColumns` | Grid dimensions (if applicable) |
| | `numComponents` | Number of distinct board components |
| | `numPlayableSites` | Number of sites where pieces may be placed |
| Topological | `avgNumDirections` | Average degree of graph vertices |
| | `avgNumOrthogonal` | Average number of orthogonal neighbours |
| | `avgNumDiagonal` | Average number of diagonal neighbours |
| Boolean flags | `isStochastic` | Game involves random outcomes |
| | `isAlternating` | Players alternate moves |
| | `hasHiddenInfo` | Game involves hidden information |
| | `isStacking` | Pieces can be stacked |
| | `hasTrack` | Board has a track (race game) |
| | `isVertexGame` | Vertex-based game |
| | `isEdgeGame` | Edge-based game |
| | `isCellGame` | Cell-based game |
| | `hasHeuristics` | Ludii metadata provides heuristics |
| | `isDeductionPuzzle` | Game is a deduction puzzle |

Games that could not be loaded reliably or produced incomplete property information were excluded from the catalog to ensure consistency and reproducibility across all experiments.

### 3.3.2 Corpus Filtering Criteria

From the full Ludii library, we restrict the experimental corpus to games satisfying the following criteria, consistent with the approach of Soemers et al. [14]:

**Two-player games only.** Multiplayer and cooperative games were excluded in order to maintain a consistent and interpretable win-rate evaluation setting.

**Deterministic games.** Games involving stochastic elements were excluded because several evaluated MCTS enhancements are primarily designed for deterministic environments, and because stochasticity introduces additional variance in performance measurements.

**Alternating-move games.** Simultaneous-move games were excluded since the MCTS framework used in this study assumes a single active player at each decision step.

**Non-deduction games.** Deduction puzzles such as Sudoku were excluded, as these problems are generally not well suited to standard MCTS approaches [14].

**Reliable game descriptions.** Games producing incomplete or unreliable property information during catalog construction were excluded to ensure consistency across feature extraction and downstream analyses.

In addition, strategies relying on heuristic evaluations were only tested on games for which compatible heuristic metadata was available within Ludii.



## 3.4 Experimental Design

This section describes the experimental protocols used to evaluate MCTS variants across the selected Ludii game corpus. The design aims to balance interpretability, computational feasibility, and coverage of diverse game characteristics.

### 3.4.1 Baseline Configuration

All experiments compare variant configurations against a fixed UCT baseline:

UCB1 | Random Playout | Monte Carlo Backprop | Robust Child

This baseline corresponds to the standard MCTS configuration most commonly used in General Game Playing research. Using a fixed reference configuration simplifies cross-game comparisons and allows observed performance differences to be attributed directly to the evaluated variants.

### 3.4.2 One-Factor-At-a-Time Design

The primary experimental protocol follows a *One-Factor-at-a-Time* (OFAT) design. In each experiment, exactly one MCTS component is modified relative to the baseline configuration, while all remaining components are kept fixed. This produces matchups of the form:

$$\underbrace{\text{UCB1}}_{\text{baseline}} \mid \underbrace{\text{MAST}}_{\textbf{variant}} \mid \underbrace{\text{MonteCarlo}}_{\text{baseline}} \mid \underbrace{\text{Robust}}_{\text{baseline}} \quad \text{vs.} \quad \underbrace{\text{UCB1}}_{\text{baseline}} \mid \underbrace{\text{Random}}_{\text{baseline}} \mid \underbrace{\text{MonteCarlo}}_{\text{baseline}} \mid \underbrace{\text{Robust}}_{\text{baseline}} \tag{3.1}$$

This design isolates the effect of individual enhancements and simplifies interpretation of performance differences. It also provides a computationally tractable alternative to exhaustive factorial evaluation, whose cost would grow combinatorially with the number of available strategies.

The OFAT study evaluates 24 individual variants spanning the selection, simulation, and backpropagation phases of MCTS. Combined with the baseline agent, this produces a

broad but interpretable comparison space suitable for large-scale evaluation across many games.

To ensure broad coverage of game characteristics, game selection prioritizes diversity within the extracted feature space. Matchups are assigned so as to maximize variation across structural and gameplay-related descriptors while avoiding duplicate evaluations.

All planning procedures were seeded with a fixed random seed to ensure reproducibility across repeated runs.

### 3.4.3 Full Combination Design



A secondary experimental setting evaluates fully configured MCTS variants in which multiple components are modified simultaneously. Unlike the OFAT protocol, this design allows interaction effects between selection, simulation, and backpropagation strategies to be studied directly.

Because the number of possible combinations grows rapidly with the number of available variants, only a strategically selected subset of combinations was evaluated. As in the OFAT setting, game selection prioritizes diversity across the extracted feature space.

### 3.4.4 Games Per Matchup and Time Budget

Each planned matchup is evaluated over 100 games, with starting positions alternated so that each agent plays equally often as Player 1 and Player 2. This reduces first-player bias and provides a more stable estimate of relative performance.

The main evaluation metric is the variant win rate against the baseline:

$$\text{WinRate} = \frac{\text{variant wins}}{\text{variant wins} + \text{baseline wins} + \text{draws}} \tag{3.2}$$

Draws are included in the denominator but count as wins for neither agent, so a win rate above 0.5 indicates that the variant outperforms the baseline.

Experiments are conducted under four per-move time budgets: 0.2, 0.5, 1.0, and 2.0 seconds. This allows the analysis to capture how variant performance changes as additional search time becomes available. The 1.0-second setting also provides a direct point of comparison with prior Ludii-based MCTS evaluations such as Soemers et al. [14].

A maximum game length of 1,000 moves is enforced across all experiments. Games that exceed this limit are terminated and recorded as draws.

## 3.5 Predictive and Statistical Analysis Framework

A central objective of this thesis is to investigate whether measurable game properties can be used to predict the relative performance of MCTS variants on previously unseen games. This is formulated as a supervised learning problem in which each game is represented by a vector of extracted structural and gameplay-related descriptors, while target variables correspond to empirically measured MCTS performance metrics.

Depending on the analysis, prediction tasks are formulated either as single-target regression problems, multi-output regression problems, or ranking-oriented evaluations aimed at identifying promising MCTS configurations for a given game. The prediction setting is inherently challenging due to the heterogeneity of the feature space, correlations between descriptors, and the stochastic noise present in empirical MCTS evaluations.

Several statistical and machine learning techniques are used to analyze the relationship between game properties and algorithmic performance.

Principal Component Analysis (PCA) is used to study the large-scale structure of the extracted feature space and to visualize similarities between games. Clustering methods are additionally used to investigate whether groups of structurally similar games exhibit comparable variant preferences.

For supervised prediction, tree-based ensemble methods (Random Forest and HistGradientBoosting) are used as primary models due to their robustness to nonlinear relationships, heterogeneous feature types, and correlated predictors. Linear baselines (Ridge and KernelRidge regression) are also evaluated to assess whether nonlinear modeling provides meaningful improvements. Both single-target and multi-output regression settings are considered.



Model evaluation relies primarily on mean squared error (MSE), mean absolute error (MAE), and $R^2$ coefficient of determination, evaluated through cross-validation on held-out test splits. Feature importance analyses are additionally used to identify which game properties contribute most strongly to predictive performance.

Each test produces one result row containing the following fields:

Example 3.2: Result row fields.

```
{testId, gameName, component,
 baselineSelection, baselineSimulation, baselineBackprop, baselineFinalMove,
 variantSelection, variantSimulation, variantBackprop, variantFinalMove,
 moveTimeSeconds, gamesPerMatchup, maxMoves, usesHeuristic,
 variantWins, baselineWins, draws, failures,
 completedGames, attemptedGames, averageMoves, lastError}
```

The primary evaluation metric is variant win rate, bounded in $[0,1]$: a value of 0.5 indicates equal performance, above 0.5 indicates the variant outperforms the baseline, and below 0.5 indicates underperformance. Failure counts and average game lengths are additionally tracked to detect pathological configurations.

## 3.6 Execution Infrastructure and Reproducibility

All experiments were executed on the CECI distributed computing infrastructure using parallel batch execution. Because individual matchups are independent, evaluations can be parallelized naturally across games, configurations, and time budgets, making the overall experimental pipeline highly scalable.

Experiments were primarily executed on multi-core CPU nodes, with large batches of independent matches distributed across the cluster scheduler. This infrastructure made it possible to evaluate thousands of game-agent matchups under multiple time controls within practical execution times.

To ensure reproducibility, experiment plans were generated deterministically and all intermediate artefacts were stored in structured tabular form. Execution logs, failed runs, and fallback behaviors were systematically recorded to facilitate auditing and debugging. Ludii's internal random number generator is seeded from the test ID string, ensuring that re-running the same test on the same machine produces identical game sequences.

The separation between planning, execution, and analysis stages additionally allows experiments to be reproduced or extended incrementally without rerunning the full pipeline.

## 3.7 Summary

This chapter introduced the experimental framework used throughout this thesis for large-scale evaluation of MCTS variants in Ludii. The framework combines a modular MCTS implementation, a systematically filtered Ludii game corpus, and reproducible large-scale evaluation protocols executed on the CECI computing infrastructure.

The experimental design includes both One-Factor-at-a-Time and multi-component combination studies, allowing the analysis of both isolated variant effects and interaction effects between MCTS components. In parallel, a predictive analysis framework was introduced to investigate whether measurable game properties can be used to model or anticipate variant performance across previously unseen games.

Together, these components provide the methodological foundation for the empirical analyses presented in the next chapter.

# 4

# Results



## 4.1 Datasets and Prediction Setup

### 4.1.1 Primary Datasets and Test Coverage

The empirical analyses in this chapter are based on two planned experiment suites, described in Chapter 3.4: a One-Factor-at-a-Time (OFAT) dataset in which a single MCTS component is varied per test, and a full-combination dataset in which all components are varied simultaneously.

Each test corresponds to a series of 100 games played under a fixed agent configuration. Table 4.1 summarizes the coverage of both datasets.

Table 4.1: Primary test coverage used in this study.

| Plan | Planned Tests | Unique Games | Time Controls (s) |
|---|---|---|---|
| One factor | 2000 | 778 | 0.2, 0.5, 1.0, 2.0 (500 each) |
| Full combinations | 2000 | 841 | 0.2, 0.5, 1.0, 2.0 (500 each) |

In the OFAT dataset, coverage is intentionally component-focused: selection variants account for 1000 tests, simulation for 435, backpropagation for 318, and final move selection for 247, reflecting the larger number of available selection strategies. In the full-combination dataset, each test evaluates a fully specified MCTS configuration drawn from the complete cross-product of available strategies.

### 4.1.2 Prediction Setup and Models

A central question in this chapter is whether the empirically measured performance of MCTS variants can be predicted from measurable game properties. This is formulated as a supervised learning problem in which the input is a fixed-dimensional feature vector representing a game-agent pair, and the target is a performance score measuring the variant's advantage over the UCT baseline.

The prediction target is defined as:

$$\text{score} = \text{winrate} + 0.5 \cdot \text{drawrate}$$

This weighted combination preserves a continuous performance signal while treating draws as intermediate outcomes, preferable to using win rate alone, which would discard information in draw-heavy games.

Two prediction formulations are evaluated: single-target regression, which predicts the score directly, and multi-output regression, which predicts win rate and draw rate separately before combining them. Both linear models (Ridge, KernelRidge) and tree-based ensemble methods (HistGradientBoosting, RandomForest) are included to assess whether the relationship between game features and MCTS performance is captured better by nonlinear approaches.

Each game-agent pair is represented by a 60-dimensional feature vector comprising static Ludii game descriptors, one-hot encoded variant identifiers, and the per-move time budget. All models are evaluated using 5-fold cross-validation on a fixed 90/10 train-test split.



## 4.2 Results

### 4.2.1 Primary Results: Model Comparison on Standard Datasets

We first evaluate whether measurable game properties can be used to predict the relative performance of MCTS variants across previously unseen games. Table 4.2 summarizes the best-performing models obtained on the two primary datasets.

Table 4.2: Best-performing model per primary dataset.

| Dataset | Best Model | MSE | MAE | $R^2$ | $n_{\text{train}}$ | $n_{\text{test}}$ |
|---|---|---|---|---|---|---|
| 1 Factor | RandomForest_MO | 0.0198 | 0.1001 | 0.6679 | 1347 | 150 |
| Full Combinations Timevar | HistGradientBoosting | 0.0288 | 0.1310 | 0.4290 | 1606 | 179 |

Across both datasets, tree-based ensemble methods consistently outperform linear models in terms of MSE, MAE, and $R^2$. However, the magnitude of this improvement remains moderate. The best-performing model on the OFAT dataset is RandomForest_MO, achieving an $R^2$ of 0.6679 (Fig. 4.1), while HistGradientBoosting performs best on the full-combination dataset with an $R^2$ of 0.4290 (Fig. 4.2).

The relative ordering of models is broadly consistent across datasets. Tree-based methods systematically outperform Ridge and Kernel Ridge regression, suggesting that the relationship between game properties and MCTS performance is at least partially nonlinear.

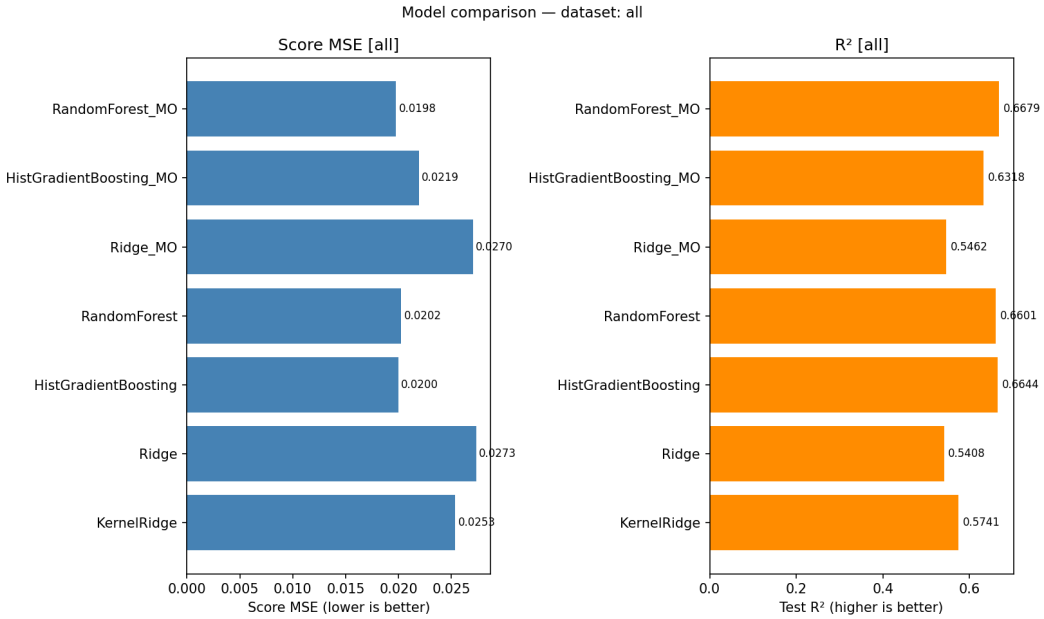


Figure 4.1: Model comparison on the OFAT dataset.

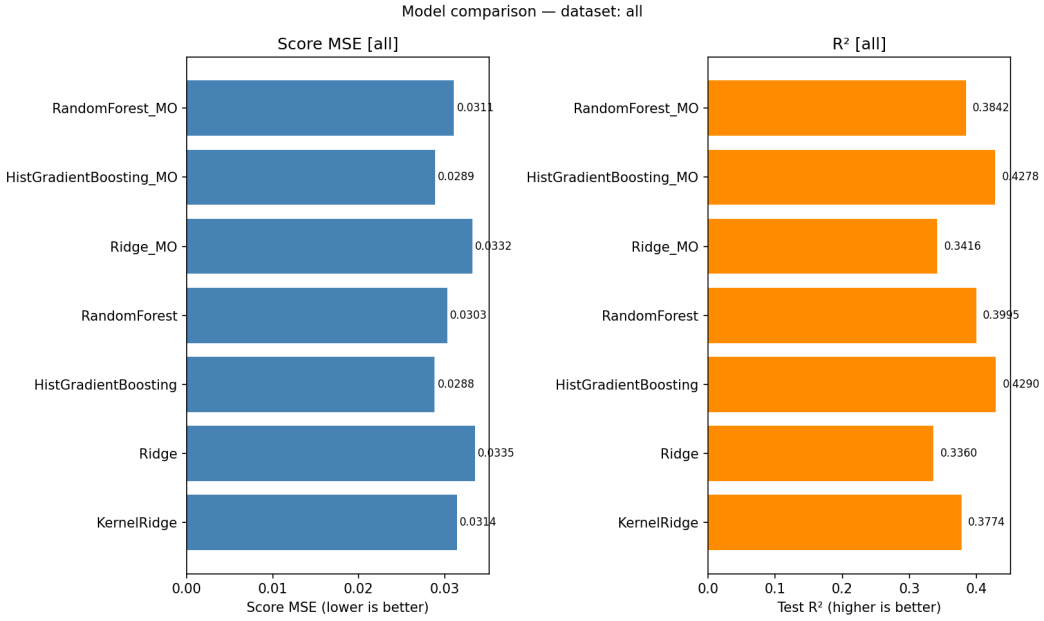


Figure 4.2: Model comparison on the full-combination dataset.

Despite these improvements, no evaluated model achieves sufficiently strong predictive performance for reliable fine-grained variant selection. Even the best-performing models retain substantial residual error, particularly on the more difficult full-combination dataset. Figure 4.3 compares predicted and true scores for the best models on each dataset.

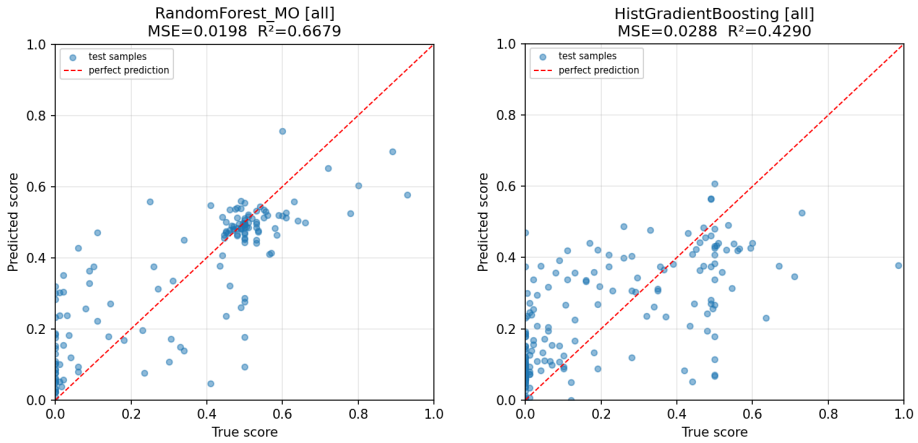


Figure 4.3: True vs predicted score for the best-performing models on each dataset. Both models capture broad global trends but retain substantial outliers in high-performance regions where ranking accuracy is most critical.

The models capture broad global trends in variant performance, but prediction errors remain large enough to prevent reliable discrimination between similarly performing configurations. This suggests the primary limitation may lie not in the predictive models themselves but in the expressiveness of the underlying game representations, which is investigated in the following sections.

## 4.3 Understanding the Limits of Prediction

The previous section showed that predictive models capture meaningful coarse trends across games, but fall short of reliable fine-grained variant discrimination. This section investigates the underlying reasons. Three complementary analyses are presented: learning curve analysis to assess data efficiency, PCA and clustering to characterize the structure of the feature space, and a cross-dataset robustness check using an independent dataset. Together, they build a coherent case that the primary bottleneck lies in the expressiveness of the available game representations rather than in the choice of predictive model or the quantity of training data.

### 4.3.1 Learning Curves and Data Efficiency

To assess whether the moderate predictive performance observed above is primarily limited by training data volume, we analyze how model accuracy evolves as larger training fractions are used. Figure 4.4 shows the validation MSE learning curves for the main model families.

Both tree-based models exhibit clear improvement at small training sizes, but the rate of improvement diminishes substantially as more data is added. HistGradientBoosting

validation MSE improves from 0.0575 at the smallest fraction to 0.0405 at the largest, while RandomForest MSE improves from 0.0490 to 0.0413, nearly plateauing over the final training fractions.

This flattening pattern indicates a transition from a data-quantity limitation to a representation limitation. If the bottleneck were purely data volume, validation loss would continue decreasing proportionally with added samples. The plateau instead suggests that the current feature representations do not contain sufficient information to distinguish MCTS performance differences more precisely, regardless of how much training data is available.

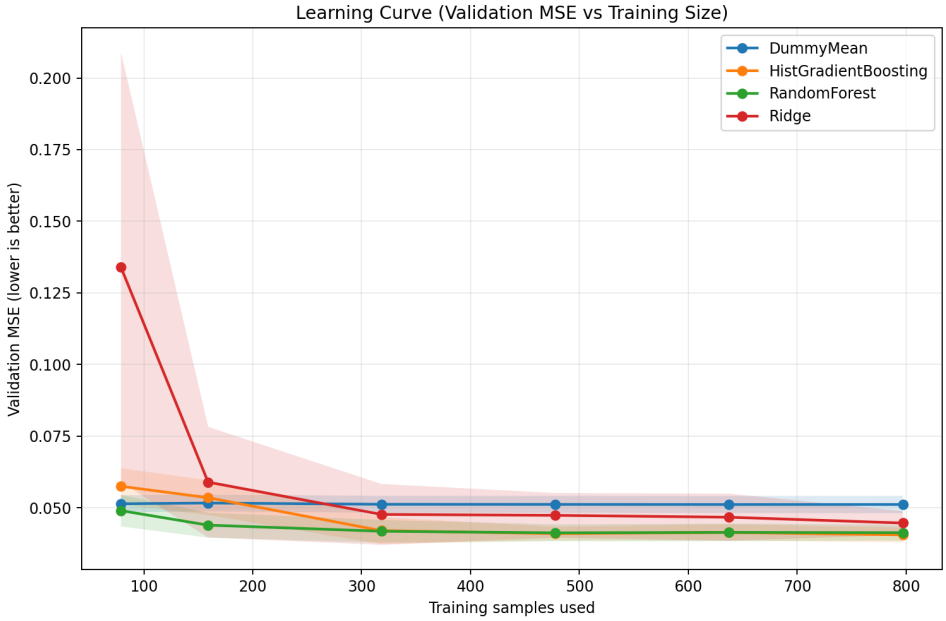


Figure 4.4: Validation MSE learning curves.

This interpretation is further supported by a targeted feature engineering experiment on the OFAT dataset. Adding two hand-crafted interaction features to the baseline feature set did not improve predictive performance and in fact caused a slight regression: MSE increased from 0.02019 to 0.02161 and $R^2$ decreased from 0.661 to 0.637. The fact that straightforward feature engineering provides no benefit, and marginally hurts performance, reinforces the conclusion that the representation bottleneck cannot be resolved by simple transformations of the existing feature set (Figure 4.5).

- Baseline: MSE=0.02019, MAE=0.10269, $R^2$=0.66080 (62 features).
- Engineered: MSE=0.02161, MAE=0.11005, $R^2$=0.63692 (64 features).

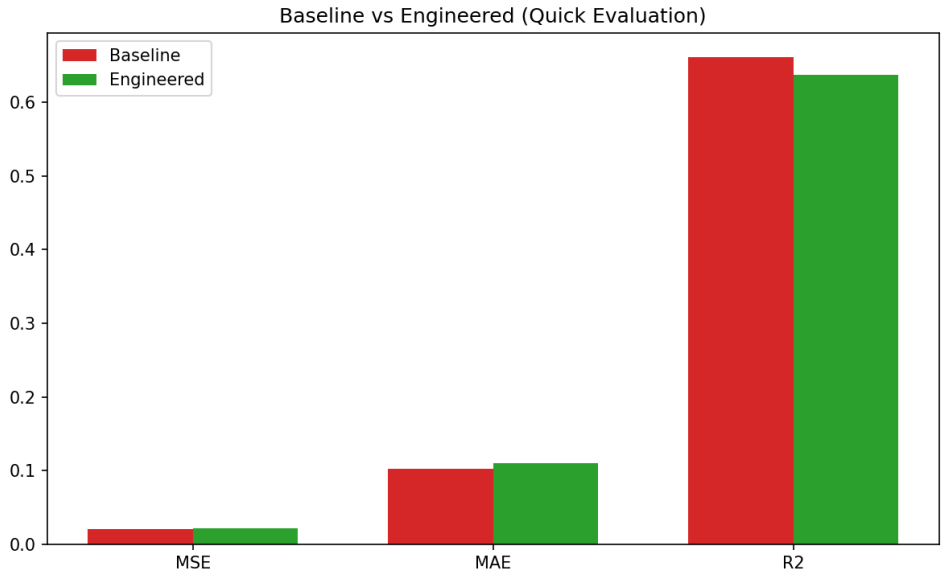


Figure 4.5: Quick feature-engineering comparison on 1f.

### 4.3.2 Feature-Space Structure: PCA and Clustering

To better understand why prediction performance plateaus, we directly analyze the structure of the game feature space. If game families cluster cleanly in feature space, and if different families exhibit distinct MCTS variant preferences, prediction should be tractable. The analyses below test whether this condition holds.

K-Means clustering was applied to game structure descriptors from the full-combination dataset, using a standard preprocessing pipeline (log1p transform, median imputation, and standardization). K-Means was selected over density-based methods because the feature space is continuous and approximately Gaussian, making centroid-based partitioning appropriate. A sweep over $K \in \{2, \ldots, 10\}$ selected $K = 2$ via silhouette score maximization (silhouette = 0.2531), as shown in Figure 4.6 (top).

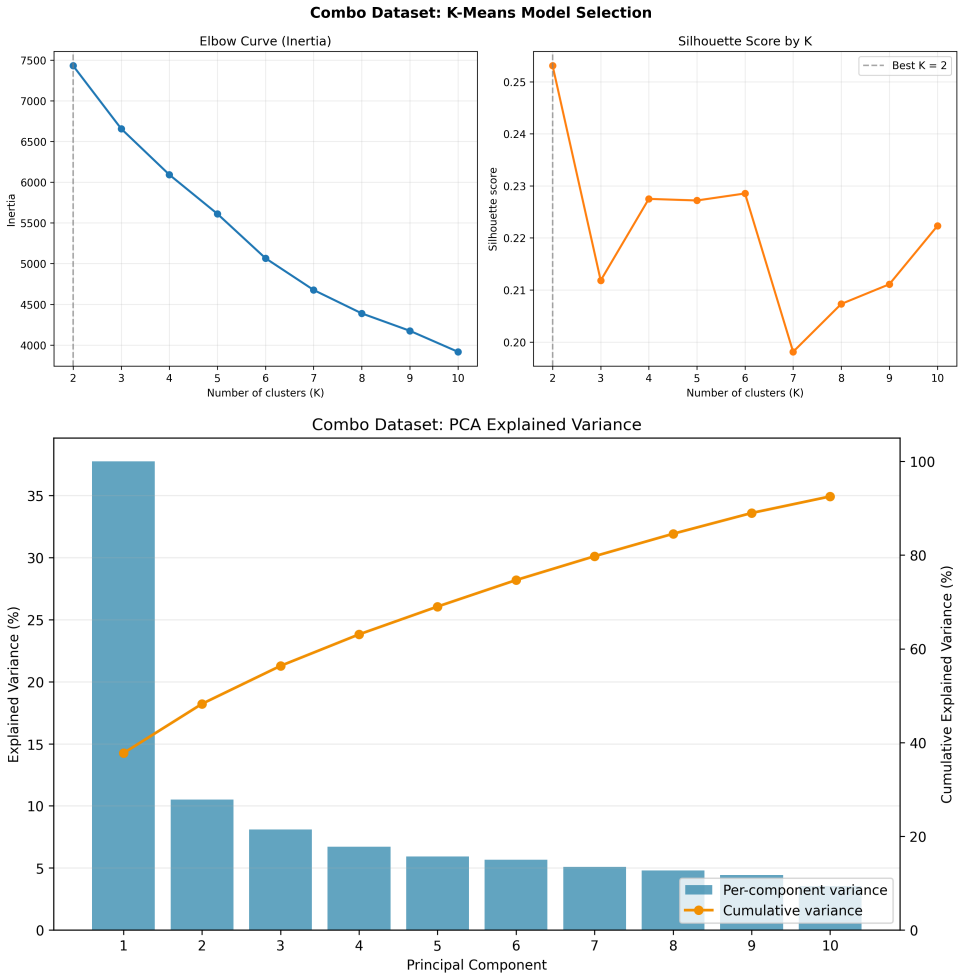


Figure 4.6: K-Means model selection (top) and cumulative explained variance (bottom). Optimal $K = 2$; first two PCs capture $< 50\%$ variance.

The two clusters correspond to interpretable game families: Cluster 0 (268 games) contains large, dense board games, while Cluster 1 (291 games) contains vertex-oriented and track-based games (Figure 4.7). This confirms that some coarse structural organization exists in the game corpus.

However, the scientific significance of this structure is limited. PCA explains only 48.26% of the total variance in the first two principal components (Figure 4.6, bottom), indicating that the feature space is intrinsically high-dimensional with no dominant low-rank structure. The PCA projection in Figure 4.8 further reveals substantial cluster overlap in 2D, meaning that no clean boundary separates the two game families in the feature space that prediction models operate on.

Adding the cluster ID as an additional feature provided only a marginal MSE improve-

ment for HistGradientBoosting (0.02881 → 0.02836), with no generalization across model families. This confirms that coarse game-family structure carries a small amount of predictive signal but is far from sufficient to resolve the representation bottleneck identified in the learning curve analysis.

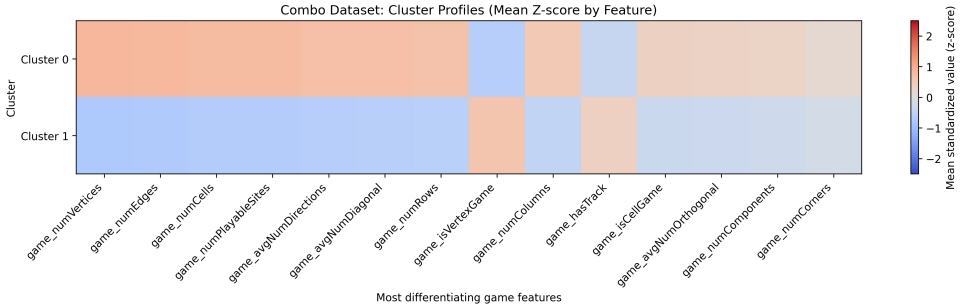


Figure 4.7: Cluster feature profiles showing distinctive game structures for each cluster.

Figure 4.8 projects games onto PC1 and PC2 colored by cluster. Despite visible cluster separation, substantial overlap remains, confirming that the feature space is too high-dimensional for clean discrimination.

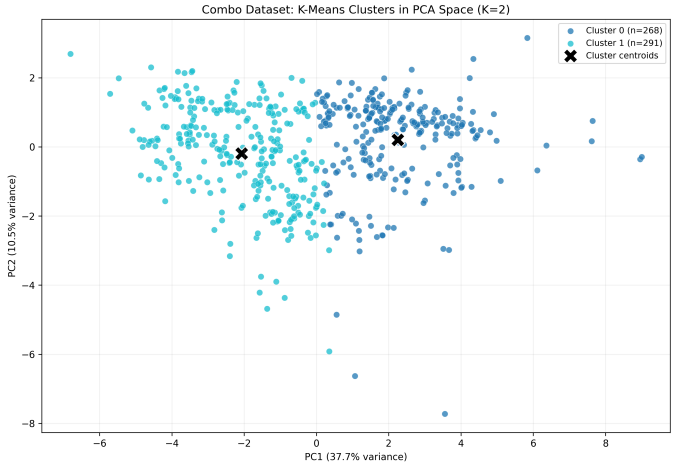


Figure 4.8: PCA projection (PC1 vs PC2) with cluster membership. Overlap is pronounced despite separation in 2D.

Adding cluster ID as a feature improved HistGradientBoosting MSE slightly (0.02881 → 0.02836), but gains did not generalize across model families. This indicates coarse game-family structure carries signal but is insufficient to overcome the representation bottleneck.

This analysis relates to prior work on measuring structural distances between board games [56], which similarly found that game feature spaces exhibit complex, overlapping

structure that resists clean low-dimensional partitioning.

### 4.3.3 Comparison with the ludii_raw Dataset

The analyses above raise the question of whether the representation limitations identified are specific to the way the OFAT and full-combination datasets were constructed. To address this, we perform a robustness check using the `ludii_raw` dataset from Stephenson et al. [16], which covers the same Ludii games but uses a different set of MCTS variants. Because the underlying feature space is identical (static game descriptors combined with variant descriptors), any patterns that persist across both datasets cannot be attributed to dataset-specific construction choices.

Table 4.3: Backup robustness snapshot (ludii_raw).

| Task | Best Model | Main Metric | Value |
|---|---|---|---|
| Regression score prediction | RandomForest | $R^2$ | 0.7571 |
| Regression score prediction | RandomForest | MSE | 0.0323 |
| Top-1 best-agent classification | Logistic Regression | Accuracy | 0.2797 |
| Top-1 best-agent classification | Logistic Regression | Macro-F1 | 0.0910 |



As seen in table 4.3, the outcome mirrors the same issues observed on our primary datasets. Even when coarse score trends are learnable (e.g., Random Forest regression achieving $R^2 = 0.7571$), the strict Top-1 best-agent prediction classification is remarkably weak (accuracy ~28%, Macro-F1 ~0.09). This robustness check solidifies our core conclusion: the current static descriptors available are simply not sufficiently discriminative for robust variant ranking across any dataset tested.

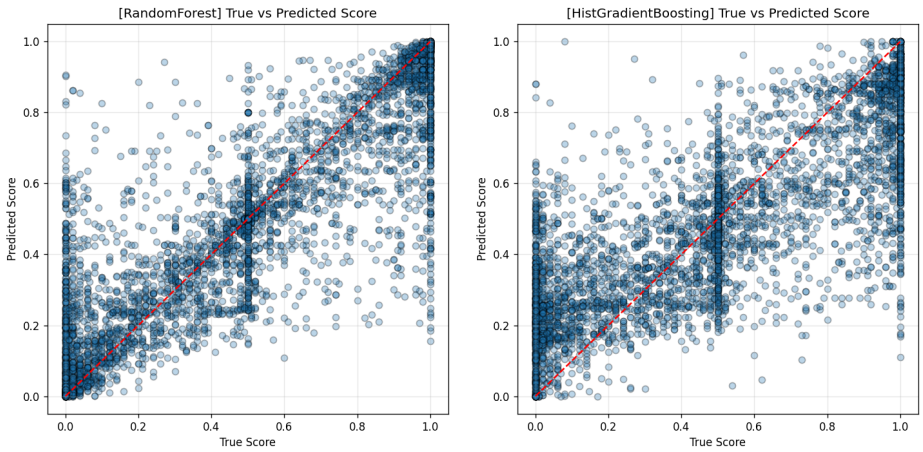


Figure 4.9: ludii_raw dataset using Random Forest (left) and HistGradientBoosting (right).

The scatter plots in Figure 4.9 reproduce the same pattern observed on the primary datasets: coarse trends are learnable but critical ranking errors persist. The consistency

of this finding across datasets with different variant sets and experimental conditions strongly supports the conclusion that the limitation is intrinsic to the current static game representations, rather than an artifact of any particular dataset. Future progress will likely require richer game representations that capture dynamic search-relevant properties beyond what static structural descriptors provide.

## 4.4 Warm-Start Transfer to Best-Agent Identification

The previous sections established that predictive models capture meaningful coarse performance trends but are not sufficiently precise for direct best-agent recommendation. However, imperfect predictions may still be useful if treated as weak informational priors rather than final decisions. This section investigates whether injecting predictive model outputs into the RCP Best Agent Identification algorithm [16] as initial estimates can improve early search efficiency, while preserving the algorithm's ability to correct incorrect predictions through online evaluation.



Every experiment is performed using the ludii_raw dataset, to best compare with the results from the original RCP paper [16]. The best-performing regression model from the previous section is used to generate predictions for each game-agent pair.

The key design question is how strongly to weight the prior. Initializing with too many pseudo-observations commits the search to potentially wrong predictions; initializing with too few discards the available signal. Two injection strategies are compared below.

### 4.4.1 Heavy Prior Initialization

The first strategy assigns $w$ pseudo-observations to the arm (variant) predicted as best by the model at the start of the RCP search, effectively pre-loading it with artificial experience. The parameter $w$ therefore controls the strength of the prior: larger values give the prediction stronger influence over exploration decisions.

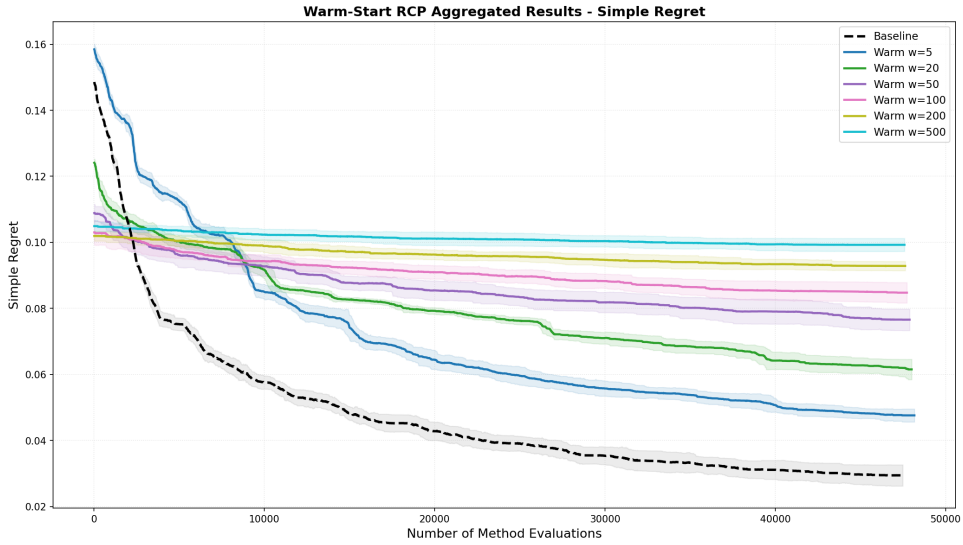


Figure 4.10: Comprehensive aggregated sweeps regret.

Figure 4.10 shows that this approach does start with an advantage, but the relatively poor prediction quality means that the search will be pulled away from the true best variant. And since a larger $w$ means more pseudo-observations, the search will be unable to course correct with any w big enough to impact the search.

### 4.4.2 Fractional Warm Start

The heavy prior initialization above provides an initial advantage but risks locking the search onto incorrect predictions when $w$ is large. As a low-commitment alternative, we propose *fractional warm start*: each arm is initialized with exactly one pseudo-observation whose value equals the model's predicted score $p$. Formally:

$$\hat{\mu}_{mk}(0) = p, \quad T_{mk}(0) = 1$$

This gives RCP a directional nudge from the predictive model while keeping uncertainty wide enough for rapid correction. Table 4.4 compares the three initialization schemes.

Table 4.4: Initialization behavior of baseline, heavy, and fractional warm starts.

| Method | Initial $\hat{\mu}$ | Initial $n$ | Initial exploration behavior |
|---|---|---|---|
| Paper RCP | 0 | 0 | Fully free exploration |
| Heavy priors ($w = 50$) | $\approx p$ | 50 | Strongly biased by prior |
| Fractional warm start | $p$ | 1 | Light directional nudge |

As shown in Figure 4.11, fractional warm start consistently reduces simple regret and average probability of error at low and medium pull counts relative to the uninformed baseline. The improvement is most pronounced in the early search phase, where the

directional signal from the predictive model helps RCP concentrate evaluation effort more quickly. At large pull counts, both methods converge to similar performance, confirming that RCP self-corrects effectively regardless of initialization.

Compared to the heavy prior ($w = 50$) from Figure 4.10, fractional warm start avoids the performance degradation caused by commitment to inaccurate predictions. This makes it the preferred initialization strategy: it captures the sample-efficiency benefits of offline learning while preserving the robustness of online adaptation. Together, these results support the broader methodological conclusion that offline prediction and online search are most effective as complementary components rather than competing alternatives.

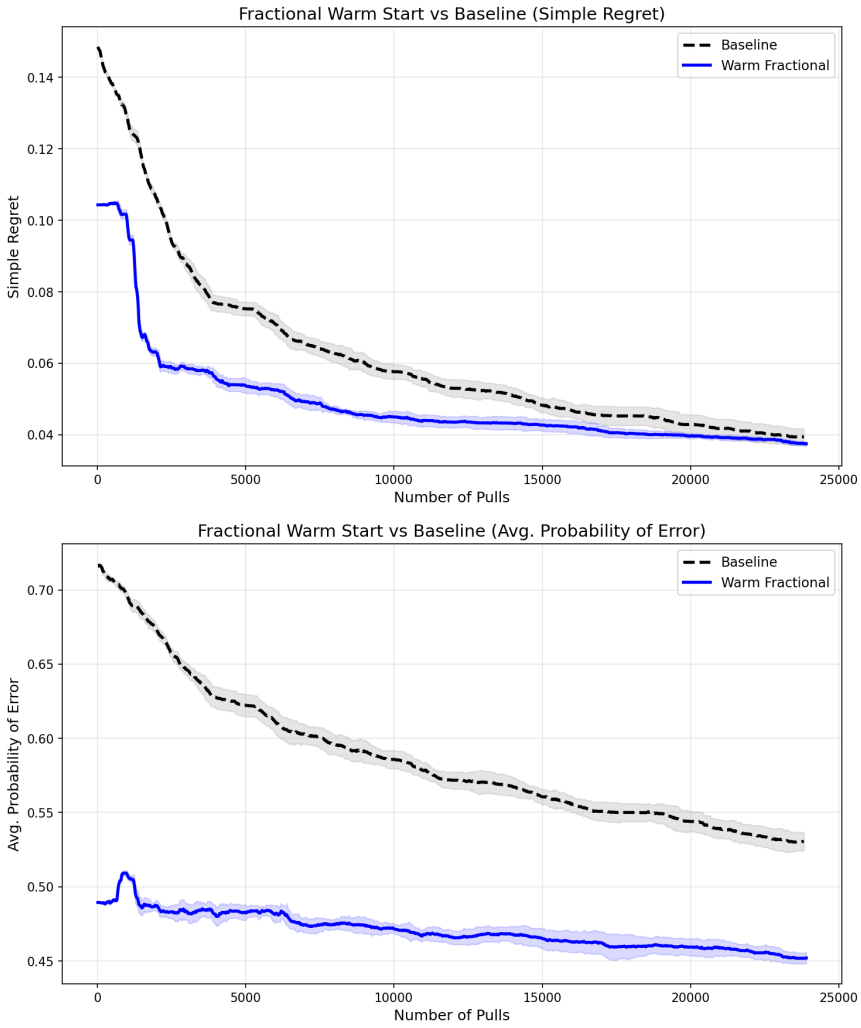


Figure 4.11: Fractional warm-start results for regret (top) and APE (bottom).

# 5

# Discussion and Implications

## 5.1 Discussion and Main Implications



The results of this thesis tell a coherent story about the current limits and practical potential of game-property-based MCTS configuration prediction.

Across both the OFAT and full-combination datasets, tree-based regressors (RandomForest, HistGradientBoosting) consistently outperform linear baselines, confirming that the relationship between game structure and MCTS performance is at least partially nonlinear. This aligns with the theoretical intuition behind Rice's algorithm selection problem [37]: problem features do carry exploitable signal for algorithm selection, and nonlinear models are better positioned to capture it.

However, the magnitude of improvement over linear baselines is modest, and no model achieves the level of accuracy required for reliable direct best-agent recommendation. The primary bottleneck, as revealed by learning curves, feature engineering experiments, and PCA/clustering analyses, appears to be the expressiveness of the available game representations rather than model capacity or data volume. The static Ludii descriptors [13] encode broad structural properties but do not capture the dynamic search-relevant characteristics, such as tactical sharpness, playout signal quality, or the concentration of game-deciding decisions, that most directly drive MCTS performance differences. This is consistent with the finding of Soemers et al. [14] that first-player advantage from random play is among the most predictive features: it is a play-derived signal, not a static structural one.

This inadequacy is not incidental but structural. Ludii's concept descriptors were designed to *characterise* and *classify* games, capturing topological and combinatorial properties, rather than to predict how a search algorithm will behave under time pressure. Two games with similar static profiles may present radically different challenges to MCTS: one may be tactically sharp, with game-deciding decisions concentrated in a narrow phase where playout noise is nearly irrelevant; another may be drawn-out and evaluation-neutral, where the balance between exploration and exploitation is critical throughout. Static features cannot discriminate these cases, and no increase in model capacity or training data will compensate for a representation blind to the distinctions that matter. For GGP specifically, where the system must generalise across a heterogeneous space of games, the feature map must be expressive along the dimensions that determine algorithmic sensitivity.

This has a direct implication for the algorithm selection paradigm as applied to GGP. Rice's framework assumes a feature space sufficiently informative about the instance-algorithm interaction [37]. When that condition fails, offline selection degrades not to mildly suboptimal recommendations but to recommendations essentially uninformative relative to the baseline of running the best average configuration. The modest gains of tree-based models over linear baselines, and the persistent gap in top-variant identification, are both consistent with a feature space only weakly correlated with the quantities that drive performance differences. The implication is pointed: reliable single-game configuration recommendation is unlikely to be achievable with static representations alone. This should be understood not as a failure of the learning approach but as a precise diagnosis of where the representational work needs to happen first.

The No Free Lunch theorem [38] provides theoretical grounding for this difficulty: in the absence of strong prior knowledge about the task structure, no fixed algorithm can universally outperform others, and the feature representations needed to distinguish "which game needs which algorithm" must be sufficiently rich to capture task-specific regularities. The current static feature set appears too coarse for this purpose.

Despite these limitations, the warm-start transfer experiments provide the most practically significant finding of the thesis. Fractional warm start reliably improves sample efficiency at low and medium evaluation budgets while preserving RCP's capacity for self-correction. This result suggests a productive reframing: rather than treating the predictive model's output as a configuration recommendation, it should be treated as a *prior distribution* over configurations, biasing initial exploration without locking out alternatives. This framing clarifies the relationship between offline learning and online adaptation: they are not competing approaches but complementary stages of a single inference process, in which offline learning narrows the prior and online search refines it. Prior work on self-adaptive MCTS [40] and algorithm portfolio methods [41] has argued for hybrid architectures on theoretical and empirical grounds; the present results provide direct evidence that even weak offline priors yield measurable efficiency gains in a GGP context.

Taken together, the results allow direct answers to the thesis's core research questions. Game structural properties *can* predict MCTS configuration performance, but only weakly and non-uniformly: predictive signal exists and tree-based models can extract it, yet the signal is insufficient for reliable direct recommendation. Offline predictions *can* improve online search efficiency, but only when applied as fractional priors rather than hard commitments, within a system that retains the capacity for self-correction.

The overall implication for GGP research is that the pursuit of offline configuration prediction should be reoriented away from direct best-agent selection and toward low-commitment initialization of online search. Predictive models are most valuable not as oracles but as informed starting points that reduce the cost of exploration.

## 5.2 Limitations

Several limitations affect the interpretation and generalizability of these results.

The most fundamental limitation concerns the game representations themselves. The features used in this study are predominantly static game descriptors derived from the Ludii API [13], capturing structural and topological properties that can be extracted without running any games. While these features provide a useful first-order characterization of

game structure, they do not capture the dynamic properties of the search space that most directly influence MCTS behavior, such as the sharpness of tactical positions, the distribution of game-deciding branching points, or the degree to which playout outcomes reflect positional quality. Incorporating such features would likely require running preliminary games or using lightweight probes, which would increase the feature extraction cost.

A second limitation concerns the prediction targets. Win rate against a fixed UCT baseline is a relative performance measure that compresses the full ranking of variants into a scalar signal. Two variants that perform very similarly to the baseline may receive nearly identical scores despite differing substantially from each other. Rank-aware objectives, such as directly optimizing Spearman rank correlation or simple regret, would be more aligned with the final use case of selecting the best variant.

Finally, the experimental scope was limited to two-player, deterministic, alternating-move games under fixed time controls. Performance patterns may differ for other game categories, longer or shorter time budgets, or against stronger baseline agents. The robustness check using `ludii_raw` provides partial evidence of cross-configuration consistency, but broader validation remains necessary.



## 5.3 Future Work

The most promising direction for future work is not a change in learning objective or model architecture, but a fundamental enrichment of the game representation. The current feature set should be extended with a broader inventory of Ludii concepts, including playout-derived features such as average branching factor, draw rate, and move-type frequencies, as well as lightweight search-derived signals that can be obtained from short preliminary game runs. Identifying which feature groups most improve best-agent identification, using ablation studies or mutual information analysis, would provide principled guidance for this effort.

A second important direction is uncertainty-aware prior injection. The warm-start experiments used fixed pseudo-count schemes regardless of prediction confidence. A more principled approach would calibrate model uncertainty and use confidence-proportional prior weights, committing more strongly to high-confidence predictions and remaining more exploratory when predictions are uncertain. This would naturally reduce the lock-in risk observed with heavy priors.

Third, the fractional warm start framework opens a natural pathway to fully adaptive hybrid systems. Combining fractional initialization with contextual bandits or online meta-MCTS methods that adjust prior strength based on observed evidence during search would allow the system to dynamically balance offline guidance and online correction throughout the evaluation budget.

Finally, once representation quality improves, the prediction models should be retrained with rank-aware loss functions, such as simple regret or top-$k$ accuracy, that are directly aligned with the best-agent identification objective rather than minimizing regression error over the full score distribution. Broader validation across additional Ludii game subsets, different time budgets, and stronger baseline agents would also strengthen the generalizability of conclusions.

# 6

# Conclusion



The central question motivating this work was whether measurable game properties can be used to predict effective MCTS configurations in General Game Playing, and whether those predictions can be turned into practical tools for accelerating Best Agent Identification.

To answer it, this thesis developed a large-scale experimental infrastructure within the Ludii General Game System and ran two complementary experiment suites, a One-Factor-at-a-Time study that isolates individual MCTS enhancements and a full-combination study that captures interaction effects, producing several thousand game-agent matchups across hundreds of structurally diverse games.

The empirical analyses reveal a consistent picture. Tree-based ensemble models outperform linear baselines across all datasets, confirming that the mapping from game structure to MCTS performance is at least partially nonlinear and that exploitable signal exists. Yet no model reaches the accuracy required for reliable fine-grained variant ranking. The bottleneck is not model capacity or data quantity: learning curves plateau well before the training set is exhausted, straightforward feature engineering yields no gain, and PCA together with clustering show only coarse, heavily overlapping structure in the feature space. Cross-dataset validation on the independent `ludii_raw` corpus reproduces the same pattern, ruling out dataset-specific artifacts.

These findings reinforce the broader challenge of adaptive search in General Game Playing: selecting effective search behavior from limited prior knowledge about a game remains difficult even when large-scale structural descriptors are available.

Despite these limitations, the experiments also show that predictive models remain practically useful when used as weak priors rather than as final decision mechanisms. Warm-start Best Agent Identification experiments demonstrate that fractional prior initialization can improve early-budget search efficiency while still allowing rapid online correction when predictions are inaccurate. This suggests that offline learning and online adaptation should be viewed as complementary components rather than competing alternatives.

The main contribution of this thesis is therefore methodological rather than predictive. It establishes that offline learning and online adaptive search are most powerful when treated as *complementary* components of a hybrid system: the predictive model provides a low-commitment starting point, and the online procedure remains responsible for robust

final selection. Pursuing increasingly complex predictive models alone is unlikely to close the remaining gap; progress will instead require richer game representations that expose the dynamic search properties the current feature set misses, tighter integration between online adaptation and offline prediction.

# Bibliography

## References


[1] Cameron Browne, Edward Powley, Daniel Whitehouse, Simon M. 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.

[2] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56:2497–2562, 2022.

[3] Mykyta Viazovskyi and Michal Certicky. StarAlgo: A squad movement planning library for StarCraft using Monte Carlo tree search and Negamax. *arXiv preprint arXiv:1812.11371*, 2018.

[4] Jorik Jooken, Pieter Leyman, Tony Wauters, and Patrick De Causmaecker. Exploring search space trees using an adapted version of Monte Carlo tree search for combinatorial optimization problems. *arXiv preprint arXiv:2010.11523*, 2020.

[5] David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems 23 (NeurIPS)*, 2010.

[6] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[7] Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2):62–72, 2005.

[8] Georgios N. Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer Nature, 2 edition, 2025. Comprehensive textbook on AI in games; 2nd edition.

[9] Eric Piette, Dennis J. N. J. Soemers, Matthew Stephenson, Chiara Sironi, Mark H. M. Winands, and Cameron Browne. Ludii – the ludemic general game system. In *Proceedings of the 24th European Conference on Artificial Intelligence*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 411–418. IOS Press, 2020.

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

[11] Cameron Browne, Dennis J. N. J. Soemers, Éric Piette, Matthew Stephenson, and Walter Crist. Ludii language reference. Technical report, Maastricht University, 2023. Technical documentation and language specification.

[12] Eric Piette, Cameron Browne, and Dennis J. N. J. Soemers. Ludii game logic guide, 2021.

[13] Eric Piette, Matthew Stephenson, Dennis J. N. J. Soemers, and Cameron Browne. General board game concepts. In *IEEE Conference on Games (CoG)*, 2021.

[14] Dennis J. N. J. Soemers, Guillaume Bams, Max Persoon, Marco Rietjens, Dimitar Sladic, Stefan Stefanov, Kurt Driessens, and Mark H. M. Winands. Towards a characterisation of Monte-Carlo tree search performance in different games. In *arXiv preprint*, volume arXiv:2406.09242, 2024. Available at `https://arxiv.org/abs/2406.09242`.

[15] 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.

[16] Matthew Stephenson, Alex Newcombe, Eric Piette, and Dennis J. N. J. Soemers. Best agent identification for general game playing. *IEEE Access*, 14:61528–61546, 2026.

[17] Stockfish Team. Stockfish. `https://stockfishchess.org/`, 2025. Free and open-source chess engine. Accessed: 2026-05-16.

[18] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018. arXiv:1712.01815.

[19] Yngvi Björnsson and Hilmar Finnsson. CadiaPlayer: A simulation-based general game player. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):4–15, 2009.

[20] Hilmar Finnsson and Yngvi Björnsson. Simulation-based approach to general game playing. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 259–264. AAAI Press, 2008.

[21] 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.

[22] 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.

[23] Cameron Browne, Eric Piette, Matthew Stephenson, and Dennis J. N. J. Soemers. Ludii general game system for modeling, analyzing, and designing board games. In *Encyclopedia of Computer Graphics and Games*. Springer, Cham, 2023.

[24] Cameron Browne, Eric Piette, Matthew Stephenson, and Dennis J. N. J. Soemers. General board geometry. In *Advances in Computer Games (ACG)*, 2021.

[25] Matthew Stephenson, Eric Piette, Dennis J. N. J. Soemers, and Cameron Browne. Ludii as a competition platform. In *Proceedings of the IEEE Conference on Games (CoG)*, 2019.

[26] Cameron Browne, Dennis J. N. J. Soemers, Éric Piette, Matthew Stephenson, Michael Conrad, Walter Crist, Thierry Depaulis, Eddie Duggan, Fred Horn, Steven Kelk, Simon M. Lucas, João Pedro Neto, David Parlett, Abdallah Saffidine, Ulrich Schädler, Jorge Nuno Silva, Alex de Voogt, and Mark H. M. Winands. Foundations of digital archæoludology. *Dagstuhl Reports*, 2019. Dagstuhl Seminar 19153.

[27] 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.

[28] Dorina Moullou, Walter Crist, Timothy Penn, and Eric Piette. Gametable network: Unveiling the past, embracing the future through ai-driven archaeological research. In *Proceedings of the European Association of Archaeologists (EAA)*, 2024.

[29] 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.

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

[31] Walter Crist, Eric 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.

[32] Eric Piette, Lisa Rougetet, Walter Crist, Matthew Stephenson, Dennis J. N. J. Soemers, and Cameron Browne. A ludii analysis of the french military game. In *Board Game Studies (BGS)*, 2021.

[33] Matthew Stephenson, Eric Piette, Dennis J. N. J. Soemers, and Cameron Browne. An overview of the ludii general game system. In *Proceedings of the IEEE Conference on Games (CoG)*, 2019.

[34] Mark J. W. Tak, Mark H. M. Winands, and Yngvi Björnsson. N-Grams and the last-good-reply policy applied in general game playing. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):73–83, 2012.

[35] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer Berlin Heidelberg, 2006.

[36] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47(2–3):235–256, 2002.

[37] John R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[38] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[39] Victor Gabillon, Mohammad Ghavamzadeh, Alessandro Lazaric, and Sébastien Bubeck. Multi-bandit best arm identification. In *Advances in Neural Information Processing Systems 24 (NeurIPS)*, pages 2222–2230, 2011.

[40] Chiara F. Sironi, Jialin Liu, and Mark H. M. Winands. Self-adaptive monte carlo tree search in general game playing. *IEEE Transactions on Games*, 12(2):132–144, 2020.

[41] Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1–2):43–62, 2001.

[42] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. On adversarial search spaces and sampling-based planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling*, volume 20, pages 242–245, 2010.

[43] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. Understanding sampling style adversarial search methods. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 474–483, 2010.

[44] Eric Piette, Matthew Stephenson, Dennis J. N. J. Soemers, and Cameron Browne. An empirical evaluation of two general game systems: Ludii and rbg. In *Proceedings of the IEEE Conference on Games (CoG)*, pages 626–629, 2019.

[45] Tristan Cazenave. Generalized rapid action value estimation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 754–760. AAAI Press, 2015.

[46] J. P. A. M. Nijssen and M. H. M. Winands. Enhancements for multi-player Monte-Carlo tree search. In *Computers and Games (CG 2010)*, volume 6515 of *Lecture Notes in Computer Science*, pages 238–249. Springer Berlin Heidelberg, 2011.

[47] Tom Pepels, Mark H. M. Winands, and Marc Lanctot. Real-time Monte Carlo tree search in Ms Pac-Man. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(3):245–257, 2014. Introduces McBRAVE; see also Pepels et al. (2014) minimizing regret.

[48] Rémi Coulom. Computing Elo ratings of move patterns in the game of Go. In *Computer Games Workshop*, 2007. Introduces progressive widening in the context of MCTS.

[49] Tom Pepels. *Monte-Carlo Tree Search is Work in Progress.* PhD thesis, Maastricht University, Maastricht, The Netherlands, 2025.

[50] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[51] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 5360–5370, 2017.

[52] Guillaume M. J.-B. Chaslot, Mark H. M. Winands, H. Jaap van den Herik, Jos W. H. M. Uiterwijk, and Bruno Bouzy. Progressive strategies for Monte-Carlo tree search. In *New Mathematics and Natural Computation*, volume 4, pages 343–357, 2008.

[53] Hendrik Baier and Peter D. Drake. The power of forgetting: Improving the last-good-reply policy in Monte Carlo Go. *IEEE Transactions on Computational Intelligence and AI in Games*, 2:303–309, 2010.

[54] Tom Pepels, Mark J. W. Tak, Marc Lanctot, and Mark H. M. Winands. Quality-based rewards for Monte-Carlo tree search simulations. In *Proceedings of the 21st European Conference on Artificial Intelligence*, 2014.

[55] Tristan Cazenave and Abdallah Saffidine. Score bounded Monte-Carlo tree search. In *Computers and Games (CG 2010)*, volume 6515 of *Lecture Notes in Computer Science*, pages 93–104. Springer Berlin Heidelberg, 2011.

[56] Matthew Stephenson, Dennis J. N. J. Soemers, Eric Piette, and Cameron Browne. Measuring board game distance. In *Computer Games (CG)*, 2022.

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