Beyond Perfect Play: A Combinatorial and Probabilistic Approach to Chess Endgame Strategy

Divij Dogra

Received August 17, 2025
Accepted October 10, 2025
Electronic access October 31, 2025

Chess is a complex game that requires deep strategic thinking, pattern recognition, calculations, and creative problem-solving. Modeling strategic decision-making in chess endgames poses a unique challenge due to the game's high complexity and the uncertainty of human play. In this paper, we propose a hybrid analytical framework that combines combinatorial game theory, graph-theoretic game-tree analysis, and probabilistic modeling to explore optimal strategies in simplified endgame positions. Using real examples including games from the 2024 FIDE World Chess Championship match between Gukesh and Ding, we construct and analyze game trees to evaluate strategies under probabilistic distributions of plausible human responses. Our approach extends traditional models by incorporating uncertainty into the decision-making process, allowing for a richer understanding of practical play in high-stakes scenarios. This framework offers a bridge between theoretical rigor and real-world applicability in chess and beyond.
Keywords: combinatorial game theory (CGT), optimal strategy, probabilistic modeling, graph-theoretic game tree analysis, chess.

Introduction

Chess has long served as a benchmark for research at the intersection of mathematics, computer science, and artificial intelligence. Its perfect-information structure, strict rules, and immense strategic depth have made it an enduring testing ground for theories of decision-making and computational search [1–6]. Yet, despite decades of progress-from Zermelo's formal proof that chess admits a determinate outcome to Shannon's analysis of game-tree complexity-most mathematical treatments of chess remain either highly abstract or narrowly focused on stylized endgames [7–10].

The central challenge lies in bridging the gap between elegant theoretical models and the complexity of real play. Combinatorial game theory (CGT), for example, provides powerful tools for reasoning about simplified endgames, where positions can often be decomposed into nearly independent subgames. Similarly, probabilistic methods have been used to approximate decision-making under uncertainty. However, existing studies typically analyze constrained cases such as king-and-pawn endings or assume perfect play, leaving open the question of how such frameworks can be adapted to positions that arise in practical, high-level competition [11].

Our work partially addresses this gap by combining CGT concepts such as game-tree representations with probability-weighted move selection. Rather than proposing an entirely new formal method, we adapt and integrate existing approaches into a hybrid framework designed for more realistic endgame scenarios. Specifically, we (i) formalize a probabilistic game-tree model in which edges are assigned likelihoods of occurrence and terminal nodes are ranked by outcome value, (ii) illustrate how pruning guided by probability weighting can reduce complexity while retaining strategic richness, and (iii) test the framework on real-world positions, including the decisive final round of the 2024 World Chess Championship [12].

In doing so, our goal is not to "solve" chess but to demonstrate how mathematical reasoning-augmented with probabilistic modeling-can bring structure and clarity to complex endgame decisions. This perspective complements both traditional CGT analysis and modern AI approaches, offering a bridge between theoretical insights and the empirical reality of competitive play.

Most relevant related work

The intersection of mathematics and chess has long attracted researchers seeking to formalize strategic insights using tools from combinatorics, game theory, and computer science. One of the earliest and most influential contributions is Ernst Zermelo's 1913 paper *Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels* [7]. Zermelo proved that chess is a finite, deterministic game of perfect information, implying that, in principle, there exists a definitive winning, drawing, or losing strategy from any given position. He addressed two foundational questions: first, how to formally characterize a winning' position; and second, whether one can determine the number of moves required to force a win. His work laid the groundwork for what would later become combinatorial game theory, and opened the door to analyzing games as structured decision trees.

Much later, Claude Shannon, in his 1950 paper *Programming a Computer for Playing Chess*, introduced the notion of game-tree complexity to quantify the challenge of solving chess by brute-force search[8]. While the number of legal positions in chess is enormous-estimated to be around 4.8×10^{44} -the number of possible games is exponentially larger. Shannon estimated that the total number of distinct move sequences (i.e., paths through the game tree) could be on the order of 10^{120}. This figure, now known as the Shannon number, illustrates the astronomical size of the search space in chess and firmly established it as a benchmark problem for early artificial intelligence research.

Later work brought these theoretical insights closer to practical gameplay by narrowing the focus to chess endgames. In *On Numbers and Endgames: Combinatorial Game Theory in Chess Endgames*, Noam D. Elkies adapts the CGT framework to simplified positions, such as king-and-pawn endings[9]. He demonstrates how certain endgame scenarios can be decomposed into nearly independent subgames, each of which can be assigned a numerical value representing the advantage of one side under optimal play. Elkies notes that such decomposition becomes more tractable in endgames because the absence of long-range pieces reduces the global interdependence between different regions of the board.

Building on this, Wu et al. (2010) in *A Combinatorial Game Theoretic Analysis of Chess Endgames* apply key CGT concepts-such as disjunctive sums, canonical forms, and outcome classes-to model specific simplified endgames[10]. Their approach uses theoretical tools to determine which player is winning, assuming perfect rationality and optimal play. However, they acknowledge that CGT becomes much harder to apply in more realistic settings due to the complexity introduced by piece mobility and interactions across the board.

More recently, advances in computer science and artificial intelligence have continued to leverage these ideas in various forms[13]. Modern engines like *AlphaZero* combine search with statistical learning to prune and evaluate the game tree more efficiently, implicitly echoing Shannon's framing while operating in vastly different computational regimes. Yet, the rigorous application of mathematical frameworks like CGT to actual high-level gameplay remains limited, in part because the tools struggle to scale beyond highly stylized or simplified positions.

Organization of the paper

The remainder of the paper is structured as follows. Section 2 introduces the fundamentals of combinatorial game theory. Section 3 presents our framework and illustrates its use through a worked numerical example on chess endgames. Section 4 discusses three additional applications in more complex settings, highlighting the versatility of the framework. In particular, we show how inputs can be derived both from expert judgment and online platforms, and we also analyze the framework's sensitivity to noise. Finally, Section 5 provides concluding remarks and outlines directions for future research and potential new applications.

Preliminaries

We start by recalling the fundamental definitions of game theory. This review of the basics provides a clear and consistent framework for our analysis, ensuring that the more advanced techniques applied later-such as combinatorial game theory and probabilistic modeling-are well grounded.

Foundations of Game Theory

Game Theory is a mathematical framework for analyzing strategic interactions-situations in which the decisions of one participant affect the outcomes for others[14 15]. A *game* is a formal model of such interactions, where players make choices with the aim of achieving outcomes they prefer. Because players often have conflicting interests, these interactions involve tension and competition, which make the study of games both rich and relevant.

Despite their diversity, all games share the following fundamental elements:

- *Players*: The decision-makers involved in the game. These can be individuals, companies, teams, political parties, or nations.
- *Choices*: At each stage, players must choose between different possible actions. A player's strategy is the rule she follows in selecting actions.
- *Outcomes*: Once all players have made their decisions, the game concludes with an outcome determined by those choices.
- *Preferences*: Each player has preferences over the possible outcomes, typically aiming to maximize her own benefit.

Game Theory also provides tools to evaluate the consequences of different choices. It allows players to assess potential gains and losses-such as whether to capture a piece or initiate an attack-and to make decisions that increase their chances of success.

Players may rely on different types of strategies. A *pure strategy* involves choosing a specific action with certainty-a fixed, deterministic plan. In contrast, a *mixed strategy* involves selecting among several actions according to a probability distribution, thereby introducing randomness into decision-making. The analysis of complex games often requires considering mixed strategies, especially when anticipating and countering an opponent's possible responses.

Representing Games Using Graphs

To analyze strategic decision-making formally, we often represent games using *graphs*, and more specifically, *trees*. This graphical representation provides a visual and mathematical structure for understanding how decisions unfold over time.

A graph G = (V, E) consists of a set of nodes V (also called vertices) and a set of edges E $\subseteq$ V $\times$ V which are connections between nodes. In a directed graph, each edge has an orientation from one node to another, representing the direction of progression-for example, from a decision point to its possible consequences. A tree is a special type of graph that is connected and contains no cycles, meaning there is exactly one simple path between any two nodes. A rooted tree designates one node as the root, from which all other nodes descend. This hierarchy induces a parent-child relationship: each node has a unique parent (except the root) and may have multiple children. The depth or layer of a node is the number of edges on the path from the root to that node. Nodes with no children are called leaves and often represent final or terminal outcomes of the process being modelled.

When we model games using trees-commonly referred to as game trees-each node represents a decision point (or game state), and each edge corresponds to an action or move taken by a player. The root node represents the initial state of the game. From there, the tree branches out according to all possible moves the players can make. As the game progresses, a path from the root to a leaf represents a complete sequence of actions-a playthrough-leading to a final outcome. The leaves thus encode the possible terminal outcomes of the game. The strategies of players can be read off the tree. A pure strategy corresponds to selecting exactly one edge (i.e., one action) at every node where a player must move. In contrast, a mixed strategy involves assigning probabilities to the different available edges-effectively introducing randomness into the decision-making process.

Given a game tree, to compute an optimal strategy, one can use a method known as backward induction. This technique starts at the leaves of the game tree-positions where the outcome is known or can be evaluated- and works backward toward the root. At each decision node, the player selects the action that leads to the best expected outcome, assuming optimal choices will be made at all subsequent steps. This recursive process continues layer by layer until it reaches the root, ultimately yielding the optimal decision at the initial state.

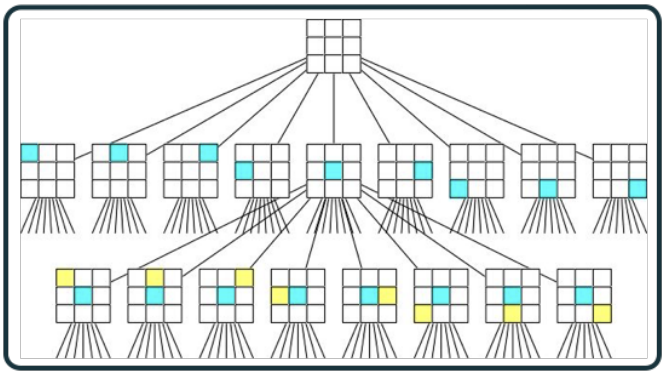

Fig. 1 A labeled game tree for the game of tic-tac-toe showing nodes as decision points, edges as actions, layers representing time steps or turns, and leaves labeled with outcomes.

Our Framework

Motivation for our framework

Chess is a paradigmatic example of a zero-sum, perfect-information game, where one player's gain is exactly the other's loss, and all elements of the game state are fully visible to both players. Every move, rule, and position is known, creating a transparent environment for analyzing strategic behavior. From a game- theoretic standpoint, this makes chess an ideal testbed: optimal strategies depend solely on players' decisions, rather than chance or hidden information.

Yet, despite this conceptual clarity, the standard tree-based representation of chess quickly becomes computationally intractable. The game's high branching factor-averaging about 31 legal moves per turn- causes the number of possible game states to explode exponentially. After only 10 moves per player, there are roughly 8.9 billion distinct positions; after 40 moves, the number of legal games approaches the astronomical Shannon number, 10^{120}. Exhaustive analysis of this tree is beyond even the most powerful computing systems.

Crucially, not all branches of the game tree are strategically relevant. Many legal moves are suboptimal, and experienced players intuitively focus on a small subset of promising moves while disregarding the rest. They anticipate opponents' responses and assign subjective probabilities, selecting moves that maximize expected outcomes rather than enumerating all possibilities.

This observation suggests a fundamental limitation of the standard game-tree-based framework: while theoretically precise, it fails to capture how human-like reasoning navigates immense strategic spaces efficiently. It motivates the development of alternative game-theoretic frameworks that move beyond exhaustive trees, emphasizing strategic relevance, probabilistic expectations, and tractable representations of complex decision spaces.

Detailed description of the model

This insight-that players implicitly filter the game tree by focusing on strategically relevant moves- motivates the need for a more compact, probabilistic representation of decision-making.

Instead of enumerating all possible moves and outcomes, our framework models only the branches that meaningfully influence the game, assigning probabilities to capture players' expectations and strategic judgments. By doing so, we create a tractable representation that mirrors human-like reasoning, blending rigorous game- theoretic analysis with practical efficiency. This approach provides a more realistic and scalable way to study complex decision-making in chess and other high-branching, perfect-information games.

We formally define our framework next. The most important notion for our framework is the one labelled game tree, an object that refines the basic game tree by partially labelling its branches as we explain next. For sake of simplicity, we restrict our attention to two player games.

Let Γ be a two-players game and the two players be denoted by P and P'. We work under the assumption that the players are rational actors whose choices aim to maximize their profit. We consider the full game trees of the game Γ that is represented by the tuple (r, V, L, E), where r denotes an initial state of the game, V is the set of all possible decision points (or game states), L is the set of the leaves of the tree representing all the possible terminal outcomes of the game, and E is the set of edges that represent available actions from one state to another.

A labelled game-tree for game Γ and player P is a tuple:

$$LGT(\Gamma, P) := ((r', V', L', E'), f, g, h)$$

where:

• The tuple (r', V', L', E') with $r' \in V, V' \subseteq V, L' \subseteq V'$,and $E' \subseteq$E is a representation of the subtree of the full game tree (r , V, L, E) of that corresponds to the portions known by player P. This subtree is rooted at r', its vertex set is V', leaves L', and its edge set is E'. As before, the nodes represent game states or decision points, leaves represents the deepest game states that the player P is considering, they do not need to be terminal outcomes of the game, and the edges represent available actions from one state to another

• The function f : V' $\rightarrow$ {P, P'}is the function that associates to each node the player who is supposed to make a choice at that point;

• The function g : $E'' \rightarrow$ [0,1] with $E'' := \{(u , v) \in E' | u \in f^{-1}(P')\}$ is the function that encodes the predictions that player P has on the decisions of player P' that he can imagine, often a strict subset of all possible decisions.

More formally, given u $\in f^{-1}(P)$, let $E_{u'}$' := {(u,v) $\in E'$ }. The function g is such that for every u $\in f^{-1}(P)$:

$$\sum_{(u,v)\in E'_{u'}} g((u,v)) = 1$$

This condition is simply encoding the fact that it induces a probability distribution on the branches corresponding to the decisions of player P'.

• h : L $\rightarrow$ {1,..., | L | } is a function that ranks the leaves from the best one (number 1) to the worst one (number |L|) in the perspective of player P.

Roughly speaking, the structure introduced partitions the nodes in two categories: the player nodes are nodes where the player corresponding to the labelled tree makes a decision; no probabilities are assigned here, as we are trying to determine the optimal move, and the opponent nodes are nodes where the opposing player makes a move; edges from these nodes are labelled with probabilities, which reflect the fixed belief (or empirical assumption) about how the opponent is likely to play. This structure results in a partial probabilistic model: it is deterministic where the player makes decisions and probabilistic where the opponent does.

In what follows, we consider the problem of identifying, given a labelled game tree for player P, an *optimal strategy*- one that *maximizes the P's expected payoff*, or equivalently, *minimizes the expected rank of the final position*. Finding a mixed strategy corresponds to assigning probabilities to the edges of the labelled decision tree, i.e., to defining a function g' : $E''' \rightarrow$ [0,1] with $E''' := \{(u , v) \in E' \mid u \in f^{-1}(P)\}$ satisfying for every $u \in f^{-1}(P)$:

$$\sum_{(u,v)\in E'_{u'}} g((u,v)) = 1$$

where $Eu' := \{(u,v) \in E'\}$. Obtaining a pure strategy from a mixed strategy is straightforward, since it suffices to take at each branching point the edge to which the mixed strategy assigned the maximum value, and assign it 1, and assign 0 to all other options.

In order to detail our strategy, we need to recall the definition of *expected value*. The expected value of a random variable is a measure of its average or mean outcome, taking into account all possible outcomes and how likely each one is. It gives a way to summarize the long-run behavior of a random process in a single number. Consider a random variable X with a *finite* list $x1, \ldots, xk$ of possible outcomes, each of which (respectively) has probability $p1, ..., pk$ of occurring. The expectation of $X$is defined as:

$$E[X] = x1p1 + x2p2 + ... + xk\,pk$$

In our setting, we consider the random variable Y that encodes the rank of the leaf reached after having applied the strategy. Let F be the family of all functions satisfying the aforementioned properties, we want to compute the function g' that minimises the value of $E(Y)$. This value is given by the following expression:

$$\sum_{l\in L'} h(l)[\sum_{\pi\in\Pi_l} w(\pi)]$$

where Πl is the set of paths from r' to l in (r', V', L', E'), and $w(\pi)$ is the product of the probabilities of the edges appearing in π. This expression contains uniquely the values of g and products of the candidate values of g' that are encoded as variables. In order to solve this problem, one can use any standard method for *constrained optimization.* Depending on the degree of the equations obtained or equivalently of the depth of (r', V', L', E'), one may need to use numerical algorithms, or branch and cut routines. In particular, one could rely on a platform such as *WolframAlpha.*

Remark. While as soon as the solver for the system is well chosen our method probably returns the best strategy with respect to the information encoded in the labelled decision tree, the quality of the performance in real world applications depends on the quality of the input data used to build the labelled decision tree. These data can have several sources such as expert judgement or data analysis, and might be affected by noise. Adding noise to data can affect the resulting strategy. This can be approximated analytically computing the derivative of the expression encoding E[X]. Since all terms are polynomial, we expect this kind of dependency.

A simple worked-out numerical example

Figure 2 represents an example of a game tree for a specific chess endgame. White has two possible moves and we assume that the player picks the move kf1. Back then has a choice of five moves as shown in figure 2. Subsequently for each of the possible black moves, white has anywhere between two to five legal moves possible.

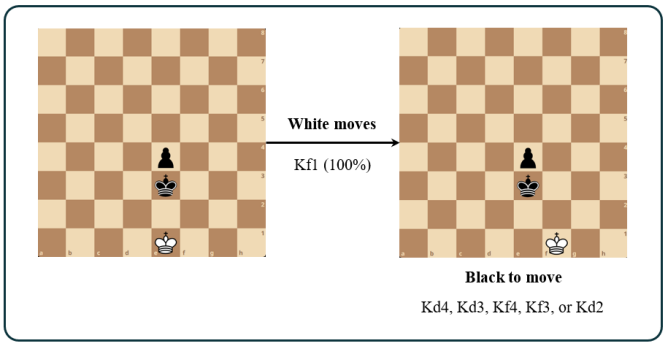

Fig. 2 A game tree representation of a specific chess end game and assuming white then moves Kf1 and subsequent potential moves for black in that scenario.

We consider the labelled game-tree given in Figure 3 which refers to the chess board position as described in Figure 2. For this simple example, we assume that the game-tree representation was given to us including probabilities for each of these edges of the tree such that for any given node the sum of these probabilities equals one. We also assume a rank value to the final nodes or leaves based on the preferences of the black player we are analyzing is given to us. In the following section, we provide concrete examples of constructions of labelled game-trees based either on expert judgement or on statistics extracted from online platforms. In this example, our objective is to compute an optimal strategy for *Black* given the predictions -encoded in the game-tree considered- for the actions of *White*.

We apply our framework to the labeled game tree shown in Figure 3. For each root-to-leaf path, we compute its probability by multiplying the probabilities of the edges along the path. Half of these probabilities are fixed numerical values, corresponding to predictions of White's actions, while the other half are variables that represent the probabilities of Black's mixed strategy, which we seek to assign in the best possible way. A solution to our problem is thus a complete assignment of these variables. Once the path probabilities are determined, we can express the expected rank of the final position reached by Black as the weighted sum of all possible rank values, where each rank is weighted by the total probability of all paths ending at leaves labeled with that rank. For the example game under consideration, this value is:

E [*final rank* (x, y, z, w)] = 1((1 − x − y − z − w) 30% + y 40%)+2(z 30%)+3(x 25%) +4(z 30%)+5((1−x −y −z −w) 70%)+6(z 40%)+7(x 50%+w)+8(y 60%)

or upon simplifying, E [*final rank* (x, y, z, w)] = 1.2x + 1.4y + 0.4z + 3.2w + 3.8

Where the variables x, y, z, w need to satisfy the following constraints:
$0 \leq x \leq 1{,}0 \leq y \leq 1{,}0 \leq z \leq 1{,}0 \leq w \leq 1{,}0 \leq 1 - x - y - z - w \leq 1$

One can then solve this system using a suitable solver such as *WolframAlpha* and see that the minimum is 21/5 and occurs at (x, y, z, w) = (0,0,1,0). Given the simplicity of the expression in this case, one could dispense with a solver and proceed as follows. Because the expected value of the final rank is linear in x, y, z, w, any minimizer occurs at an extreme point of the *feasible set*, the set of points that satisfies the constraints introduced. The extreme points of the feasible set are the vectors with three coordinates equal to 0 and the remaining one equal to 1 (i.e. the standard basis vectors), since the constraint x+y+z+w=1 together with the box constraints forces vertices to be 01 vectors with exactly one 1. Therefore, it suffices to assign to 1 the variable with the lowest coefficient.

Next we perform a sensitivity analysis by perturbing the probabilities for potential moves by white by +/-5%, +/-10%, +/-15%, and +/-20% as shown in Figure 4 and recalculate the expected values of the final rank.

With various path probabilities as shown in Figure 4, we can again express the expected ranks of the final position reached by Black as the weighted sum of all possible rank values, where each rank is weighted by the total probability of all paths ending at leaves labeled with that rank for each set of the sensitized probability columns. For the example game under consideration,

Figure 3: Example of labelled game-tree for a specific chess endgame.

Black move	**Black Probability**	**White move**	**White Probability**	**End Position Rank (Black perspective)**
Kd2	x	Kf2	50%	7
Kd2	x	Kg2	25%	3
Kd2	x	Kg1	25%	3
Kd4	y	Ke1	10%	1
Kd4	y	Ke2	60%	8
Kd4	y	Kf2	10%	1
Kd4	y	Kg2	10%	1
Kd4	y	Kg1	10%	1
Kd3	z	Ke1	40%	6
Kd3	z	Kf2	30%	4
Kd3	z	Kg2	15%	2
Kd3	z	Kg1	15%	2
Kf3	w	Ke1	50%	7
Kf3	w	Kg1	50%	7
Kf4	1-x-y-z-w	Ke2	35%	5
Kf4	1-x-y-z-w	Ke1	10%	1
Kf4	1-x-y-z-w	Kf2	35%	5
Kf4	1-x-y-z-w	Kg2	10%	1
Kf4	1-x-y-z-w	Kg1	10%	1

Figure 4: Example of labelled game-tree for a specific chess endgame with sensitivity analysis for white's move probabilities.

Black move	**Black Prob**	**White move**	**White Prob**	**End Position Rank (Black perspective)**	**Sensitivity White Prob 5%**	**Sensitivity White Prob 10%**	**Sensitivity White Prob 15%**	**Sensitivity White Prob 20%**
Kd2	x	Kf2	50%	7	54%	59%	63%	67%
Kd2	x	Kg2	25%	3	22%	18%	15%	12%
Kd2	x	Kg1	25%	3	24%	23%	22%	21%
Kd4	y	Ke1	10%	1	22%	35%	44%	51%
Kd4	y	Ke2	60%	8	55%	51%	46%	41%
Kd4	y	Kf2	10%	1	9%	9%	8%	7%
Kd4	y	Kg2	10%	1	7%	4%	2%	0%
Kd4	y	Kg1	10%	1	6%	1%	0%	0%
Kd3	z	Ke1	40%	6	47%	54%	61%	68%
Kd3	z	Kf2	30%	4	27%	24%	21%	18%
Kd3	z	Kg2	15%	2	12%	8%	5%	2%
Kd3	z	Kg1	15%	2	14%	14%	13%	12%
Kf3	w	Ke1	50%	7	52%	55%	57%	60%
Kf3	w	Kg1	50%	7	48%	45%	43%	40%
Kf4	1-x-y-z-w	Ke2	35%	5	43%	51%	59%	67%
Kf4	1-x-y-z-w	Ke1	10%	1	8%	7%	5%	3%
Kf4	1-x-y-z-w	Kf2	35%	5	30%	25%	21%	16%
Kf4	1-x-y-z-w	Kg2	10%	1	9%	7%	6%	5%
Kf4	1-x-y-z-w	Kg1	10%	1	10%	10%	9%	9%

we get the following:

E [+/-5% *final rank* (x, y, z, w)] $= 1.2x + 0.9y + 0.5z + 3.1w + 3.9$

E [+/-10% *final rank* (x, y, z, w)] $= 1.3x + 0.5y + 0.6z + 2.9w + 4.1$

E [+/-15% *final rank* (x, y, z, w)] $= 1.3x + 0.0y + 0.7z + 2.8w + 4.2$

E [+/-20% *final rank* (x, y, z, w)] $= 1.4x - 0.4y + 0.8z + 2.7w + 4.3$

Where the variables x , y, z , w need to satisfy the following

constraints:
$0 \leq x \leq 1{,}0 \leq y \leq 1{,}0 \leq z \leq 1{,}0 \leq w \leq 1{,}0 \leq 1 - x - y - z - w \leq 1$

One can then solve this system using a suitable solver such as *WolframAlpha* and see that the minimum for the first expression occurs at $(x, y, z, w) = (0,0,1,0)$, however, the minimum changes to (0,1,0,0) for the expressions where probability was perturbed by more than +/-10%.

Applications of our framework

We now apply our probability weighted game tree analysis to three different chess games including the 2024 World Championship classical chess final game between Grand Master (GM) Gukesh Dommaraju and Grand Master (GM) Ding Liren and two additional chess end games.

FIDE World Chess Championship: GM Gukesh vs. GM Ding

Organized by the International Chess Federation (FIDE), World Chess Championship is the premier event to determine the world champion in classical chess. The world chess championship is typically a best-of-14 games series. It's a match between the reigning world champion and a challenger, typically the winner of the Candidates Tournament, a separate event where top players compete for the right to challenge the reigning champion. The winner of the match earns the title of World Chess Champion.

After an incredibly hard-fought game, it suddenly ended when the Chinese reigning champion GM Ding Liren blundered in the nal classical game. We will look at a specific point in this final game to determine an optimal chess strategy using our framework and compare it to the actual player moves that proved to be a turning point in chess history.

In the chess board positions shown in Figure 5, Ding Liren (playing for white) moved Rf2 which put him at a significant disadvantage! We apply our framework beginning with Gukesh's next move (playing black) and we analyze the various possibilities in the game following Ding's last move for Gukesh such that it maximizes black's opportunity to win the position.

For our analysis, we assign probabilities to each of the nodes corresponding to Ding's subsequent moves based on the views from an expert, my chess coach who is a National Master in chess. We also calculate the probabilities using the Stockfish score assigned by the LiChess software. A negative Stockfish score indicates that the game is in favor of the Black player and a positive Stockfish score indicates that the game is in favour of the White player. In order to convert these scores to probabilities, we use the function *exp(*A.L*)*, where L is the Stockfish score and A is a scaling factor (we used A = 0.1 for our analysis). We then assign a rank to the end position from Gukesh's perspective

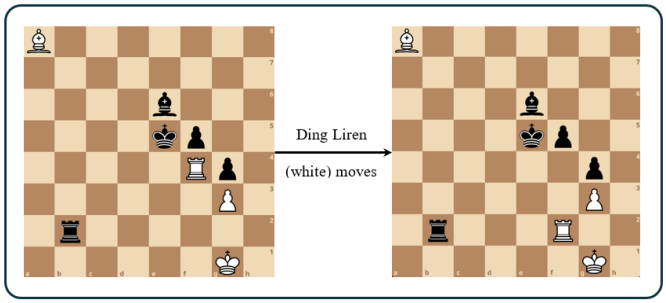

Figure 5: Board positions from the final game of 2024 FIDE World Chess Championship, showing moves before and after GM Ding Liren blundered.

based again on the views from my chess coach, as illustrated in Figure 6.

We calculate the probability of each path by simply starting with the rank of the end position and multiplying it with the probabilities assigned to each node in that path. We then sum up all these expected values for each of the possible paths to get a cumulative E (final rank). For the chess game position above, using chess field expert assigned probabilities, we get an equation for Expected Value of the end position rank from Ding's perspective to be:

E [*final rank* (x, y, z, w)]= $-4.7x + 1.7y + 2z + 1.9w + 5.7$

If we use the probabilities calculated using the Stockfish score instead, the expression becomes:

E[*final rank* (x, y, z, w)]= $-3.9x + 2.1y + 1.5z + 2.1w + 4.9$

Next we optimize these E[final rank] to find the minimum in order for Gukesh to get a rank as close to the top (1) as possible using *WolframAlpha*, which gives us EV = 1 for both equations, when $(x, y, z, w) = (1,0,0,0)$. We highlight this strategy in the game tree chart in green which represents an optimal path by minimizing the EV. This path also reflects the actual game played during the championship highlighting that after Ding's sub optimal move of Rf2, Gukesh played the optimal strategy to maximize his chances to win the championship game.

Another example: Pawn end game

We analyze another chess game with the following board position as shown in Figure 7, where black player has four likely moves.

We assign variables *x, y, z,* and *1-x-y-z* as probabilities for these four moves and then evaluate probabilities of white players' next move for each based on the Stockfish score from LiChess software. As shown in Figure 8, we then rank these end positions from Black players perspective and using our framework, set up an expression for Expected Value by calculating each path:

E[*final rank* (x, y, z)]= $-0.6x + 3.8y + 2.8z + 3.1$

Figure 6: A game tree representation of the moves after GM Ding Liren blundered final game of 2024 FIDE World Chess Championship, showing: five possible moves for GM Gukesh, subsequent potential moves for GM Ding in each of the scenarios with probabilities assigned, and the associated end position rank from GM Gukesh's (black's) perspective.

Gukesh (Black) move	***Stockfish Scores***	**Black Prob**	**Ding (White) move**	***Stockfish Scores***	**White Prob Expert view**	**End Position Rank (Black perspective)**	**White Prob calc from Stockfish**
Rxf2	*-30.0*	x	Kxf2	*-28.0*	100%	1	100%
Rb1+	*-0.3*	y	Rf1	*-34.8*	10%	1	2%
Rb1+	*-0.3*	y	Kh2	*-1.1*	20%	5	47%
Rb1+	*-0.3*	y	Kg2	*-0.2*	70%	9	51%
Rb8	*-0.1*	z	Bc6	*-0.3*	20%	7	26%
Rb8	*-0.1*	z	Bg2	*-0.3*	10%	5	26%
Rb8	*-0.1*	z	Bh1	*-2.2*	10%	4	22%
Rb8	*-0.1*	z	Re2+	*-0.2*	60%	9	26%
Ra2	*0*	w	Rxa2	*0*	55%	9	34%
Ra2	*0*	w	Bc6	*-0.2*	30%	7	33%
Ra2	*0*	w	Bg2	*-0.4*	10%	5	32%
Ra2	*0*	w	Bh1	*-35.3*	5%	1	1%
Rb3	*-0.2*	1-x-y-z-w	Kg2	*-0.4*	40%	6	29%
Rb3	*-0.2*	1-x-y-z-w	Kh2	*-3.3*	10%	3	22%
Rb3	*-0.2*	1-x-y-z-w	Re2+	*-0.2*	40%	7	30%
Rb3	*-0.2*	1-x-y-z-w	Rg2	*-4.7*	10%	2	19%

Figure 8: A game tree representation of another chess game showing: four possible moves for the black player, subsequent potential moves for the white player in each of the scenarios with probabilities assigned based on Stockfish scores from LiChess software, and the associated end position rank from the black player's perspective.

Black move	**Black Prob**	**White move**	**White Prob**	**End Position Rank (Black perspective)**	**Prob weighted rank**	**Stockfish Scores after black move**	**Stockfish Scores after white move**
b3	x	axb3	48%	2	1.0	-28	-19
b3	x	cxb3	52%	3	1.6	-28	-18
a3	y	bxa3	97%	7	6.8	35.5	36
a3	y	b3	3%	5	0.1	35.5	0
c3	z	bxc3	94%	6	5.7	17.2	28
c3	z	b3	6%	5	0.3	17.2	0
Kg5	1-x-y-z	b3	20%	1	0.2	-96	-21
Kg5	1-x-y-z	Kg3	27%	3	0.8	-96	-18
Kg5	1-x-y-z	a3	54%	4	2.1	-96	-11

Another example: Tight corner endgame

We now analyze yet another chess game with the following board position as shown in Figure 9, where black player has four likely moves.

We assign variables *x, y, z,* and *1-x-y-z* as probabilities for these four moves and then evaluate probabilities of white players' next move for each based on the Stockfish score from LiChess software. As shown in Figure 10, we then rank these end positions from Black players perspective and using our framework, set up an expression for Expected Value by calculating each path:

$$E[\textit{final rank}\ (x, y, z)] = -2.0x + 0.2y + 1.0z + 3.0$$

Concluding remarks

Chess is a complex game and modeling strategic decision-making for the game poses a unique challenge given uncertainty of human play. In this paper, we proposed a hybrid analytical framework that combined combinatorial game theory, graph-theoretic game tree analysis, and probabilistic modeling to ex-

Figure 10: A game tree representation of another chess game showing: four possible moves for the black player, subsequent potential moves for the white player in each of the scenarios with probabilities assigned based on Stockfish scores from LiChess software, and the associated end position rank from the black player's perspective

Black move	**Black Prob**	**White move**	**White Prob**	**End Position Rank (Black perspective)**	**Prob weighted rank**	**Stockfish Scores after black move**	**Stockfish Scores after white move**
Bd4#	x	-	100%	1	1.0	-1	-15
Kxg4	y	Kxg2	22%	2	0.4	-12	-11
Kxg4	y	Kh2	45%	4	1.8	-12	-4
Kxg4	y	Kf2	33%	3	1.0	-12	-7
Kf3	z	Kh2	100%	4	4.0	-5	-4
Kh3	1-x-y-z	Kf2	100%	3	3.0	-8	-7

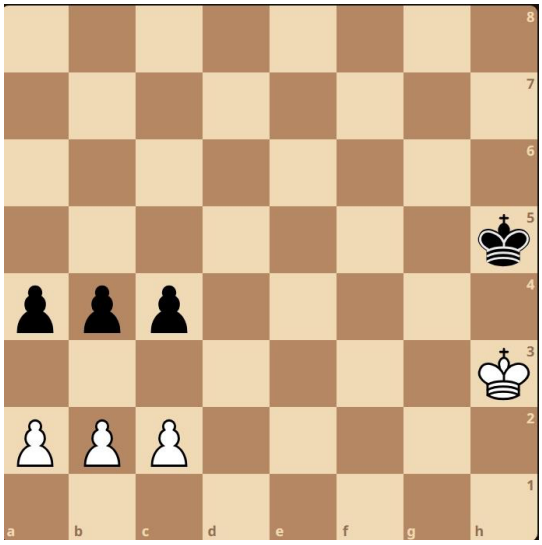

Figure 7: Initial board position for a Pawn end game

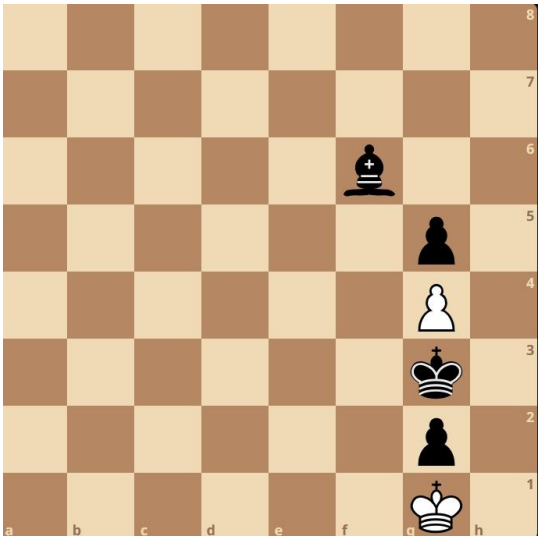

Figure 9: Initial board position for a Tight corner endgame.

plore optimal strategies in select chess positions. We started with the analysis of a simple chess endgame to illustrate our framework and then extended it to a few other examples including one from a recent real example from the 2024 FIDE World Chess Championship match between Gukesh and Ding. In our analyses, we constructed and analyzed game trees starting with specific chess game positions and evaluated strategies under probabilistic distributions from different sources. For the example from the 2024 FIDE World Chess Championship, we also compared the actual game moves with the optimal strategy determined by our framework.

This framework can be a useful training tool for chess players to evaluate and improve their game strategies. It can also be extended to model real-world decisions beyond chess, such as in business scenarios where a set of decisions may lead to different strategic paths and have a potential measurable financial impact. Indeed, we described our framework in terms of 2-player games and not chess.

The main limitation of this work at present is the need to "manually" prune the game tree to keep the analysis tractable. However, the framework holds potential for significant enhancement through the use of software programming and artificial intelligence tools. Future research could focus on automating the assignment of probabilities to different branches and implementing rank assignment after a given number of moves to improve scalability and precision. Notably, this issue of limited data does not arise in the context of chess, where widely used platforms like Lichess provide rich statistical data that can be readily integrated into such frameworks. However, such transparency is not yet evident in many other applications, where access to relevant data remains limited or entirely unavailable.

Acknowledgments

I would like to express my sincere gratitude to my research mentor, Gaia Carenini, who is a PhD student at the Department of Pure Mathematics and Statistics at the University of Cambridge, for her invaluable guidance, patience, constructive feedback, and sustained support throughout the duration of this work. I am also grateful to my parents for their encouragement and steadfast belief in my academic pursuits. My thanks extend as well to my research project coordinator for their exceptional organization skills and commitment, which greatly facilitated the progress of this research study.

References

1 R. Levinson, F. Hsu, T. Marsland, J. Schaeffer and D. Wilkins, *Panel: The role of chess in artificial intelligence research*.

2 F. Gaessler and H. Piezunka, *Training with AI: Evidence from chess computers*.

3 M. Bilali, M. Graf and N. Vaci, *Computers and chess masters: The role of AI in transforming elite human performance*.

4 D. Tosi and M. Scalise, *Int. Workshop Hybrid Models for Coupling Deductive and Inductive Reasoning*, Springer Nature Switzerland, Cham, p. 2234.

5 M. Maschler, S. Zamir and E. Solan, *Game theory*, Cambridge University Press.

6 A. Tucker, *Mathematician answers chess problem about attacking queens*, `https://www.quantamagazine.org/mathematician-answers-chess-problem-about-attacking-`.

7 E. Zermelo, *Uber eine Anwendung der Mengenlehre auf die Theorie des Schachspiels*.

8 C. Shannon, *Programming a computer for playing chess*.

9 N. Elkies, *On numbers and endgames: Combinatorial game theory in chess endgames*, Games of No Chance, p. 345370.

10 W. Wu, Y. Yu and Z. Cai, *A combinatorial game theoretic analysis of chess endgames*.

11 J. Ypma, *Chess endgame analysis: Reinforcement learning vs. search tree algorithms*.

12 F.I.D.E., *Chess World Championship Gukesh vs*, `https://www.chess.com/events/2024-fide-chess-world-championship/14/Ding_Liren-GukeshD(2024)`.

13 G. Marcus, *Google DeepMind trains artificial brainstorming in chess AI*, `https://`.

14 M. Osborne and A. Rubinstein, *A course in game theory*, MIT Press.

15 R. Myerson, *Game theory: Analysis of conflict*, Harvard University Press.

