On Optimal Strategies for Wordle and General Guessing Games

Michael Cunanan and **Michael Thielscher**

School of Computer Science and Engineering, University of New South Wales
cunananm2000@gmail.com, mit@unsw.edu.au

Abstract

The recent popularity of Wordle has revived interest in guessing games. We develop a general method for finding optimal strategies for guessing games while avoiding an exhaustive search. Our main contributions are several theorems that build towards a general theory to prove the optimality of a strategy for a guessing game. This work is developed to apply to any guessing game, but we use Wordle as an example to present concrete results.

1 Introduction

Mastermind is a guessing game that has been studied extensively in the past [Knuth, 1977; Stuckman and Zhang, 2005; Doerr *et al.*, 2016; Glazik *et al.*, 2021]. Such work has not seemed to be carried over to other guessing games, however. Our vision is to have AI agents learn how to approach any game of this kind, similar to a general-game-playing setting [Genesereth and Björnsson, 2013; Genesereth and Thielscher, 2014]. To do this we supply human intelligence to guide this area of research; this paper aims to do just that for general guessing games. We also aim to add mathematical rigour to the study of meta-reasoning in guessing games, such as in [Filman *et al.*, 1983], or to aid in developing predicates for grounded languages such as in [Thomason *et al.*, 2016].

The timing of this publication coincides with the recent popularity of the online game Wordle [Wardle, 2021], which we will use for our example guessing game of choice. Wordle is a word game that was published in October 2021. Since then, it has gained significant popularity, with over 300,000 daily users in January 2022 [Serrels and Boom, 2022]. There has been widespread interest in the general community for an optimal approach to the game, with several websites making unsupported claims to have determined the best strategy.

The game itself is a guessing game in which players must deduce a hidden word using clues that the game gives in response to the player's guesses, with a fixed limit of 6 guesses allowed. The exact details of these clues and the structure of the game will be explored in further detail in the next section.

The popularity of Wordle has also caused several variants to appear, including with

- Different word sets (e.g. Bardle [Bardle, 2022], FFXIV-rdle [FFXIV, 2022])

- Multiple games at the same time (e.g. Dordle [Dordle, 2022], Tridle [Tridle, 2022], Sexaginta-quattuordle [Sexaginta-quattuordle, 2022])

- Completely different forms of input (e.g. Heardle [Heardle, 2022], Chessle [Chessle, 2022]).

As such, the focus of this paper lies in guessing games in general, but we will use Wordle as the main example throughout.

Our main contribution is a series of theorems that build towards a general method to determine if a strategy is optimal or not, without the need for an exhaustive search. These formal results can also be used to find an optimal strategy. The theorems we present are generalized to work for any guessing games to automatically find strategies and prove their optimality. We specifically demonstrate using these theorems to show the Wordle strategy found by our framework is optimal. We also present a method of determining the next optimal guess, which to our knowledge, is a novel approach.

The remainder of the paper is organized as follows. In the next section, we recapitulate the basic components of guessing games in general, including Wordle, and we recapitulate known heuristics from the literature on Mastermind. In Section 3, we show how to combine heuristics to search for good strategies. In Section 4, we present novel and general theorems by which a strategy can be proved optimal without an exhaustive search. In the section that follows, we demonstrate using the general method and theorems on Wordle and some of its variants. We conclude in Section 6.[1]

2 Background

2.1 Guessing Games

In this section, we define exactly what we consider to be general guessing games, following similar definitions by Koyama and Lai [1991] and Focardi and Luccio [2012]. Koyama and Lai refer to a guessing game as an 'interactive knowledge transfer model', but for the sake of readability we will use the term 'guessing game'.

In a **guessing game**, we have two parties: a **learner** and a **teacher**. The teacher's goal is to communicate some **secret** s that is initially hidden from the learner. The learner submits a **guess** g to the teacher, to which the teacher responds

---

[1]An extended version of this paper with full proofs and more examples can be found on arXiv: http://arxiv.org/abs/2305.09111.

with some **response** r. The teacher's responses are to be used as clues by the learner to deduce what s is. The teacher computes responses using an **answering function** a; this function is known to both parties. These communications continue until the teacher responds with the **affirmative response** r^*, at which point we say the learner has learnt the secret and has won the game. The following definition summarizes the components of a general guessing game.

Definition 1 (Guessing game). *A guessing game can be uniquely represented as a tuple* (G, S, R, r^*, a), *where*

$G = $ *Set of allowable guesses*

$S = $ *Set of allowable secrets, with* $S \subseteq G$

$R = $ *Set of possible responses, with* $|R| > 1$

$r^* = $ *Affirmative response, with* $r^* \in R$

$a = $ *Answering function of type* $G \times S \to R$,

$\qquad$ *with* $\forall g \in G \, \forall s \in S : a(g, s) = r^* \iff g = s$

and G, S *and* R *are finite sets.*

We assume that S is known to the learner, though exactly which element of S is the secret is not known. This may not be true in practice for human players, but we do this because any guessing game should have a well-defined domain of secrets that an AI certainly could use.

2.2 Wordle

In Wordle[2], the player (learner) must deduce a common 5-letter English word chosen by the computer (teacher). Similar to the well-known guessing game Mastermind, the player's guesses are met with colour-coded responses to guide them toward the answer. The secret in Wordle changes daily.

We provide an example play of Wordle in Figure 1 using the game from March 22 2022. The player's first guess was TARES. The computer assigned a grey colour to T, A, R, E, and so none of those letters appear in the secret word. The letter S however was assigned a yellow colour, which indicated that S does appear in the secret, but not in its current position (i.e. the 5th position). The player's next guess was SPOIL. Now S and O are assigned green, which indicates S and O appear in those positions in the secret word. The player uses these colour encodings as responses from the computer to determine what to guess next throughout the game. Wordle ends when the player receives an all-green encoding, as seen in the final row.

To represent colours we use $0 = $ Grey, $1 = $ Yellow and $2 = $ Green in the following definition.

Definition 2 (Wordle). *Wordle is a guessing game* $(G_W, S_W, R_W, r_W^*, a_W)$ *according to Definition 1, where*

$G_W = $ *All 5 letter words, curated by the developer.*

$S_W = $ *Common 5 letter words decided by the developer.*

$R_W = \{00000, 00001, \ldots, 22220, 22222\}$

$r_W^* = 22222$

$a_W = $ *See the above paragraph and Figure 1 for example.*

[2]Wordle has several variants. In this paper we consider Wordle in its default mode.

Figure 1: March 22 2022 Wordle puzzle, completed in 4 turns.

G_W and S_W are publicly known sets and can be found in Wordle's source code. It is worth noting however that this set has been altered a few times since the game's creation. Our research into Wordle strategies was initially conducted on Wordle's original sets of guesses and secrets (before 15th February 2022), and so we describe our work using these sets, with $|G_W| = 12972$ and $|S_W| = 2315$. Importantly however, the results presented in this paper could easily be replicated for the updated Wordle sets, or any other word set in general, as we will see.

2.3 Strategies

In this section, we formalize how we intend agents to play guessing games by defining strategies; as well as how we intend to compare the performance of strategies.

As a learner plays a guessing game, they should be using the previously submitted guesses and the corresponding received responses to make informed decisions about what guess to submit next. We may capture this learned information using **candidates**.

Definition 3 (Candidates). *Suppose in a guessing game* (G, S, R, r^*, a) *the guesses and responses so far are* $((g_1, r_1), \ldots, (g_n, r_n))$, *i.e. guess* g_i *was met with response* r_i. *The **candidate set** C is defined as*

$$C = \bigcap_{i=1}^{n} \{s \in S : a(g_i, s) = r_i\}$$

If there are no guesses or responses so far, $C = S$.

Candidates are elements of S that *could* be the secret word according to the information contained in the guesses and responses played so far. At the start of a game, the set of candidates is S as no information about the secret has been communicated to the learner yet.

A **strategy** is a learner's method of determining what guess to submit next, recalling the goal is to get response r^*. We make use of candidate sets in formalising this notion.

Definition 4 (Strategy). *A strategy σ can be defined as*

$$\sigma : P(S) \to G$$

where $P(S)$ is the power set of S. Then $\sigma(C)$, for some candidate set $C \subseteq S$, would represent what guess to submit next.

Note that the co-domain of σ is G; the learner is allowed to make guesses that may not be possible secrets. Strategies also may be non-deterministic, but in this paper, we only consider ones that are deterministic.

On receiving a response from the teacher, we need to filter the candidate set appropriately. After submitting a guess, we can know the possible future candidate sets; in this sense each guess can **split** the candidate set.

Definition 5 (Split). *For a candidate set C, we say that guessing g creates **splits** categorized by response r:*

$$C_{g,r} = \{c \in C : a(g,c) = r\}$$

We can calculate our score playing according to any strategy using TURNSNEEDED.

Definition 6 (TURNSNEEDED). *Suppose we play using strategy σ and the hidden secret is s. Start with candidate set $C = S$ and submit guess $g = \sigma(C)$. If response r^* is received, then we are done. Otherwise, replace C with $C_{g,r}$ and again submit guess $\sigma(C)$. Repeat until response r^* is received. TURNSNEEDED(σ, s) is the number of guesses submitted.*

Note that in this process, the player does not use s to decide what to guess; we only use the remaining candidates and σ to determine what to guess next.

The objective of this paper is to find an 'optimal' strategy. Existing papers measured the performance of strategies by taking the EXPECTED number of guesses needed (taken over all secrets in S) [Koyama and Lai, 1994; Focardi and Luccio, 2012]. Other authors such as Kooi [2005] also considered the maximum number of guesses needed, but the primary goal historically has always been to minimize the EXPECTED score. In this paper we will be using an equivalent metric TOTAL as defined below.

Definition 7 (TOTAL metric). *For a strategy σ, TOTAL(σ) is the total number of turns needed over all secrets in S:*

$$\text{TOTAL}(\sigma) = \sum_{s \in S} \text{TURNSNEEDED}(\sigma, s)$$

It should be clear that a strategy that is optimal according to the EXPECTED case is also optimal according to the TOTAL $(= |S| \cdot \text{EXPECTED})$ case. We use TOTAL however because it makes the work in Section 4 much easier to read.

2.4 Known Strategies

In the extensive literature on Mastermind [Knuth, 1977; Bestavros and Belal, 1986; Kooi, 2005; Berghman *et al.*, 2009], several strategies have been developed and tested. We restate some of these strategies in this section for later reference. They all determine what guess to submit next by assigning each guess g a numerical score based on the current candidate set using a **valuation**.

Definition 8 (Valuation-based strategy).

$$\sigma_v(C) = \underset{g \in G}{\arg\min}\, v(g, C)$$

*where C is a candidate set and v is some function of type $G \times P(S) \to \mathbb{R}$. We call v a **valuation**. In tie-breaks, default to lexicographical ordering.*

A simple (yet useful) valuation is the following:

$$\text{INSET}(g, C) = -\mathbb{I}[g \in C]$$

where $\mathbb{I}$ is the indicator function. This is adapted from one of the earliest published algorithms on Mastermind [Sterling and Shapiro, 1994]. We use the negative sign since we are taking the min in Definition 8, and prioritising guesses that *are* in C.

Strategies developed for Mastermind focused on using different valuations such as:

$$\text{MAXSIZESPLIT}(g, C) = \max_{r \in R} |C_{g,r}|$$

$$\text{EXPSIZESPLIT}(g, C) = \sum_{r \in R} \left(\frac{|C_{g,r}|}{|C|} \cdot |C_{g,r}| \right)$$

$$\text{INFORMATION}(g, C) = \sum_{r \in R} \frac{|C_{g,r}|}{|C|} \log_2 \frac{|C_{g,r}|}{|C|}$$

$$\text{MOSTPARTS}(g, C) = -\text{NSPLITS}(g, C)$$

where

$$\text{NSPLITS}(g, C) = |\{r \in R : |C_{g,r}| \neq 0\}|$$

Knuth [1977] used MAXSIZESPLIT, Kooi [2005] introduced MOSTPARTS and Bestavros and Belal [1986] used INFORMATION. There are several other valuations developed for Mastermind, but we only included the ones which showed promising results in existing literature.

3 Finding Good Strategies

Before we can prove the optimality of a strategy for any given guessing game, it is necessary to first find a *good* one. We do this by first using Knuth's [1977] paper on Mastermind for inspiration, and revisit Definition 7 to develop a method by which we may search for strategies with low TOTAL scores.

3.1 Combining Known Valuations

Knuth [1977] used the MAXSIZESPLIT evaluation. In the event several guesses had equally minimal valuations, he suggested that for the next guess, "a valid one should be used", i.e. a guess that is also a candidate. He made no explicit rule about which to choose if there are multiple guesses with equally minimal valuations **and** that are candidates. We resolve this in the context of a general guessing game.

Definition 9 (Combined valuations). *For valuations $v_1, \ldots, v_n$, we can combine them to assign each guess g a tuple of values:*

$$V(g, C) = (v_1(g, C), \ldots, v_n(g, C))$$

We can then compare tuples lexicographically.

Only if the combined valuations together are the same for two guesses, *then* we may revert to choosing alphabetically, but ideally we would append more valuations to avoid this.

In choosing which valuations to combine for Wordle, we tested[3] every non-empty ordered combination of INSET, MAXSIZESPLIT, INFORMATION, MOSTPARTS, EXPSIZESPLIT, giving 325 combined valuations.

---

[3]Full source code for this experiment and all subsequent ones is available at https://github.com/cunananm2000/WordleBot.

Rank	Combined Valuations	TOTAL
1	MOSTPARTS,INSET,ESS	7944
2	MOSTPARTS,INSET,ESS,MSS	7944
3	MOSTPARTS,INSET,MSS,ESS	7944
⋮	⋮	⋮
323	MSS	8510
324	INSET,MSS	8516
325	INSET	10069

Table 1: Combined valuations on Wordle. For space, we shorten EXPSIZESPLIT to ESS and MAXSIZESPLIT to MSS.

As shown by Table 1, using combined valuations does offer an improvement over using any single valuation alone. Moreover, all the best-performing combinations use MOST-PARTS as the main valuation, which makes sense as it was the best-performing standalone valuation for Mastermind [Kooi, 2005].

3.2 Searching For Strategies

Koyama and Lai [1991] presented an equation to calculate the minimum EXPECTED score achievable. We adapt this to an equation to calculate the minimum TOTAL score:

Definition 10 (MINTOTAL). *For a non-empty candidate set C, the minimum* TOTAL *number of guesses needed to reach all candidates in C is given by*

$$\text{MINTOTAL}(C) = |C| + \min_{g \in G} \sum_{r \in R \setminus \{r^*\}} \text{MINTOTAL}(C_{g,r})$$

If C is empty, then $\text{MINTOTAL}(C) = 0$.

Unfortunately, it isn't feasible to calculate MINTOTAL for most candidate sets in real guessing games such as Wordle. The recursive definition means that with each calculation of MINTOTAL we loop over $g \in G$. If we were to limit our recursion depth to d, then our algorithm would run in $O(|G|^d)$. To limit this exponential growth, we propose that instead of searching over all $g \in G$, only search over the 'best' n guesses in G; we call n the **search breadth**.

Definition 11 (Approximate MINTOTAL). *For a candidate set C, the **approximate** minimum total number of guesses needed is* $\text{APMINTOTAL}(C, n)$, *defined by replacing $g \in G$ in Definition 10 with $g \in \{Best\ n\ guesses\}$.*

We will be using the topmost combined valuation from Table 1, (MOSTPARTS, INSET, EXPSIZESPLIT), to determine what the best n guesses are; taking the n guesses with the lowest valuations.

Depending on how exhaustively we want to look for strategies we may change n; a higher value of n means a more exhaustive search. It should be clear, then, that $\text{APMINTOTAL}(C, n) \geq \text{MINTOTAL}(C)$ for any $n \geq 1$, and that at $n = |G|$ the two are equal.

As mentioned previously, we may use the argmin of the 'otherwise' case of Definition 11 to extract a strategy.

Since Wordle is the main guessing game for this paper, we first show our results in Table 2. As we expect, the TOTAL

n	APMINTOTAL(S_W, n)	Starter
1	7944	TRACE
5	7921	SALET
10	7920	SALET
20	7920	SALET

Table 2: Using APMINTOTAL to find a good strategy for Wordle.

Game	$|G|$	$|S|$	APMINTOTAL($S, 20$)
FFXIVrdle	849	168	432
Mininerdle	206	206	544
Nerdle	17723	17723	53512
Primel	8363	8363	29011

Table 3: Using APMINTOTAL to find a good strategy for other popular guessing games, for which we purposely show the EXPECTED rather than TOTAL. The choice of breadth 20 was due to Table 2.

decreases as the search breadth increases, as this means we search more exhaustively.

We repeated this process of using Definition 11 to search for strategies with low TOTAL scores on the following variants of Wordle:

- **FFXIVrdle**: $S =$ References to the video game Final Fantasy 14, e.g. `HILDA`.
- **Mininerdle**: $G = S = 6$ character math equations, e.g. `4*7=28`.
- **Nerdle**: $G = S = 8$ character math equations, e.g. `8*3+2=26`.
- **Primel**: $G = S = 5$ digit prime numbers, e.g. `42821`.

Results are shown in Table 3.

4 Proving Optimality

The previous section was focused on using heuristics to find good strategies; now we'd like to determine if the best ones found were indeed optimal. First, we revisit Definition 10, and explore the idea of representing strategies as 'trees'. Doing so allows us to prove several propositions which we use to create novel theorems by which we can prove a strategy optimal *without exhaustive search*.

4.1 Useful Guesses

In order to help restrict the search space, we define the notion of *usefulness*.

Definition 12. *For a candidate set C with $|C| > 1$, a guess $g \in G$ is **useful** w.r.t. C iff* $\text{NSPLITS}(g, C) \neq 1$. *We notate this as $g \in \text{UG}(C)$ for short. If $|C| \leq 1$, then $\text{UG}(C) = C$.*

Property 1. *Equivalently for $|C| > 1$, $g \in \text{UG}(C)$ iff $|C_{g,r}| < |C|$ for all $r \in R$.*

Lemma 1. *For any non-empty $C \subseteq S$,*

$$\text{MINTOTAL}(C) = \min_{g \in \text{UG}(C)} |C| + \sum_{r \in R \setminus \{r^*\}} \text{MINTOTAL}(C_{g,r})$$

Note the replacement of $g \in G$ from Definition 10 with $g \in \text{UG}(C)$.

Proof. Suppose the minimum was achieved by some $g \notin \text{UG}(C)$, so $C_{g,r} = C$ for a specific r and $C_{g,r'} = \emptyset$ for $r' \in R \setminus \{r\}$. By Definition 10 this would imply $\text{MINTOTAL}(C) = |C| + \text{MINTOTAL}(C)$, so $|C| = 0$, contrary to the assumption. $\qquad\square$

This also shows that optimal strategies can only have useful guesses.

4.2 Setup

Definition 13 (V^*)**.** *By splitting Lemma 1, we define*

$$\text{MINTOTAL}(C) = \min_{g \in \text{UG}(C)} V^*(g, C)$$

$$V^*(g, C) = |C| + \sum_{r \in R \setminus \{r^*\}} \text{MINTOTAL}(C_{g,r})$$

and $\text{MINTOTAL}(\emptyset) = 0$.

$V^*(g, C)$ then represents the minimum TOTAL, starting from candidate set C, **provided** we guess $g \in G$ first. The **optimal** strategy would then be achieved by taking the argmin.

$\text{MINTOTAL}(C)$ and $V^*(g, C)$ are what we should try to estimate. Finding an upper bound for $\text{MINTOTAL}(C)$ is easy; as noted previously $\text{APMINTOTAL}(C, n)$ is an upper bound for any n. We notate such an upper bound as $\text{UB}(C)$. As per Table 2, the lowest known value found for Wordle, $\text{UB}(S_W)$, is 7920. Lower bounding V^* is important via the following theorem:

Theorem 1. *Suppose we have some function* UB *such that* $\text{UB}(C) \geq \text{MINTOTAL}(C)$ *for any* $C \subseteq S$*. If we can find some estimate function* V' *such that* $V'(g, C) \leq V^*(g, C)$ *for any guess* $g \in G$ *and* $C \subseteq S$*, then for any* $g' \in G$

$$V'(g', C) > \text{UB}(C) \implies g' \neq \underset{g \in \text{UG}(C)}{\arg\min} V^*(g, C)$$

Proof. If $V'(g', C) > \text{UB}(C)$ for a particular $g' \in G$,

$$V^*(g', C) \geq V'(g', C) > \text{UB}(C) \geq \text{MINTOTAL}(C)$$

We then use Definition 13 to complete the proof. $\qquad\square$

Theorem 1 has the effect that for any guess g', if $V^*(g', S) > \text{UB}(S)$, then g' **cannot** be an optimal starting word.

In order to estimate V^*, we must first estimate MINTOTAL as it is much easier to create bounds for.

4.3 Tree Representations

Ville [2013] demonstrated representing their Mastermind strategy as a decision tree, and we may do the same with strategies in guessing games in general, as illustrated in Figure 2 for a Wordle strategy. We will call these **strategy trees**. Each node is a guess to be submitted; starting at the root node as the initial guess. The outgoing branches from a node represent the possible responses received by submitting the node's guess. If r^* is a possible response, then we do **not** include that branch and instead highlight the node in green as a possible end to the game.

Alternatively, each node can be thought of as corresponding to a current set of candidates C, labelled with $\sigma(C)$.
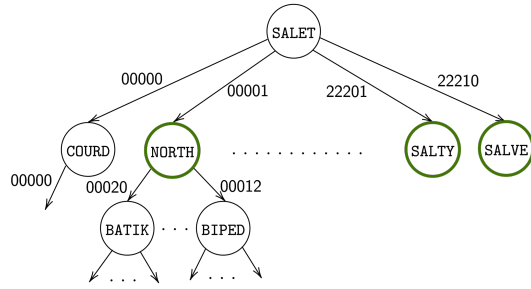

Figure 2: Part of our Wordle strategy represented as a strategy tree. Not all branches or guesses are shown.

It follows that for the tree representation of a strategy σ, the value $\text{TURNSNEEDED}(\sigma, s)$ is represented by the depth of node labelled with s *as a leaf node*. Note that the same word may appear multiple times in a strategy tree, so we must follow the nodes and branches to properly compute the 'correct' depth. Importantly, we assign the depth of the root node as 1, so the value $\text{TOTAL}(\sigma)$ can be then visualized as the sum of the depths of each secret. With this, we can then notice that MINTOTAL is purely dependent on the placements of the nodes corresponding to possible secrets within a tree. The natural question to ask then is, "What is the best way to arrange the nodes corresponding to possible secrets in a strategy tree to minimize the sum of depths to each of these nodes?", or put more generally,

"What is the best way to arrange the n nodes in a tree to minimize the sum of depths to each node?"

To answer this we need to prove some properties about strategy trees.

Definition 14. *For any* $C \subseteq S$*,*

$$\text{MAXSPLITS}(C) = \max_{g \in G} \text{NSPLITS}(g, C)$$

Lemma 2. *For candidate sets* C' *and* C*, if* $C' \subseteq C$*, then* $\text{MAXSPLITS}(C') \leq \text{MAXSPLITS}(C)$*.*

Proof. If $C' \subseteq C$, for any guess $g \in G$ we must have $\text{NSPLITS}(g, C') \leq \text{NSPLITS}(g, C)$. This is because if $C'_{g,r}$ is non-empty for some response r, then $C_{g,r}$ is non-empty. This implies the desired result. $\qquad\square$

Theorem 2. *In any tree made to resolve a candidate set* C*, all nodes have at most* $\text{MAXSPLITS}(C)$ *children.*

Proof. The value $\text{MAXSPLITS}(C)$ is the highest number of branches the root node can have. This is true even after noting that r^* is never assigned a corresponding branch. Recall moreover that each child node also corresponds to a candidate set $C' \subset C$. The number of children that the direct child nodes of the root node can have is upper bounded by $\text{MAXSPLITS}(C')$, but by Lemma 2, this value is upper bounded by $\text{MAXSPLITS}(C)$. The same logic can be cascaded down each branch of the tree to show that each node has at most $\text{MAXSPLITS}(C)$ children. $\qquad\square$

Definition 15. *For integers $n \geq 0$ and $b \geq 1$, BOUND (n, b) is the minimum sum of depths of each node in a tree with n nodes, and each node having at most b children. We call such a tree a b-tree.*

Theorem 3. *For integers $n > 0$ and $b > 1$,*

$$\text{BOUND}(n, b) = \sum_{i=1}^{k} i b^{i-1} + (k+1) \cdot \left(n - \frac{b^k - 1}{b - 1} \right)$$

where $k = \lfloor \log_b(n(b-1) + 1) \rfloor$

For $b = 1$, BOUND$(n, 1) = \frac{n(n+1)}{2}$.
For $n = 0$, BOUND$(0, b) = 0$.

Proof. Note that BOUND$(0, 1) = 0$ by either of the last two cases.

This is trivial when $n = 0$ or $b = 1$. In the case where $n > 0$ and $b > 1$, there are multiple ways to arrange n nodes in a b-tree. We are interested in minimizing the sum of depths to each node; clearly we must fill in level-order, noting that there are at most b^{i-1} nodes of depth i. Doing this will show that k is the depth of the last completely filled layer. The last term in the definition accounts for the 'leftover' nodes at depth $k + 1$. $\square$

The restriction on the number of children suggests we use BOUND in creating lower bounds for MINTOTAL.

Definition 16.

$$LB_1(C) = \text{BOUND}(|C|, \text{MAXSPLITS}(S))$$
$$LB_2(C) = \text{BOUND}(|C|, \text{MAXSPLITS}(C))$$

In turn, we use this to recursively build bounds for V^* and MINTOTAL, taking inspiration from Definition 13.

Definition 17. *For integers $i \geq 1$*

$$LB_{i+2}(C) = \min_{g \in \text{UG}(C)} V_i(g, C)$$

$$V_i(g, C) = |C| + \sum_{r \in R \setminus \{r^*\}} LB_i(C_{g,r})$$

It remains to prove that LB_i and V_i are in fact lower bounds to MINTOTAL and V^* respectively.

4.4 Key Theorems and Proofs

We now use the work of Sections 4.1, 4.2 and 4.3 to build the key theorems of this paper.

Lemma 3. *For any $C \subseteq S$, $LB_1(C) \leq LB_2(C)$.*

Proof. Note that BOUND (n, b) increases as b decreases for a fixed n. Recall the definition of BOUND (n, b). Clearly decreasing b means each node's depth can only increase, so the overall sum of depths for each node must increase.

This fact combined with Lemma 2 implies the desired result. $\square$

Lemma 4. *For any $C \subseteq S$, $LB_2(C) \leq$ MINTOTAL (C).*

Proof. MINTOTAL is intended to represent the best way to arrange the candidates of C in any valid strategy tree in order to minimize the sum of depths to each candidate. We know that in this 'ideal' strategy tree, there must be at least $|C|$ nodes (one for each candidate), and that by Theorem 2 this tree is a MAXSPLITS(C)-tree. It follows by Definition 16 and Definition 15 that the sum of depths to each node is lower bounded by $LB_2(C)$. $\square$

Lemma 5. *If $LB_i(C) \leq LB_j(C)$ for any $C \subseteq S$, then $V_i(C) \leq V_j(C)$ for any guess $g \in G$ and $C \subseteq S$.*

Proof. Follows from construction in Definition 17. $\square$

Proposition 1. *If $V_i(C) \leq V_j(C)$ for any guess $g \in G$ and $C \subseteq S$, then $LB_{i+2}(C) \leq LB_{j+2}(C)$ for any $C \subseteq S$.*

Proof. Follows from construction in Definition 17. $\square$

Corollary 1. *If $LB_i(C) \leq LB_j(C)$ for any $C \subseteq S$, then $LB_{i+2}(C) \leq LB_{j+2}(C)$ for any $C \subseteq S$.*

Proof. Follows from Lemma 5 and Proposition 1. $\square$

Theorem 4. *For any integer $n \geq 1$, we have that $LB_{2n-1}(C) \leq LB_{2n}(C) \leq$ MINTOTAL (C) for any $C \subseteq S$.*

Proof. We may follow a similar proof to Corollary 1 to show that if $LB_i(C) \leq$ MINTOTAL (C) for any $C \subseteq S$, then $LB_{i+2}(C) \leq$ MINTOTAL (C) for any $C \subseteq S$.

The desired result follows combining this with Lemmas 3 and 4 along with Corollary 1. $\square$

Theorem 5. *For any integer $n \geq 1$, we have that $LB_n(C) \leq LB_{n+2}(C) \leq$ MINTOTAL (C) for any $C \subseteq S$.*

Sketch Proof. We can show that for any $C \subseteq S$, we have $LB_1(C) \leq LB_3(C)$. Intuitively, $LB_1(C)$ is the minimum sum of depths in a tree assuming that each node has at most MAXSPLITS (S) children. $LB_3(C)$ however asserts that the root node **must** split according to a legitimate guess. This restriction implies $LB_1(C) \leq LB_3(C)$.

We can also show that $LB_2(C) \leq LB_4(C)$, for any $C \subseteq S$. Proving this uses the previous claim of $LB_1(C) \leq LB_3(C)$. It requires the trick that we may replace S in the explicit definition of that claim with C, since we only require that S be a super-set of C.

Combine these two inequalities with Lemmas 3 and 4, and Corollary 1 to reach the desired conclusion. $\square$

Proposition 2. *For any integer $n \geq 1$, $V_n(g, C) \leq V^*(g, C)$ for any guess $g \in G$ and any $C \subseteq S$.*

Proof. Theorem 4 shows that $LB_n(C) \leq$ MINTOTAL (C) for any integer $n \geq 1$ and any $C \subseteq S$.

By the construction of V^* (Definition 13) and V_i (Definition 17), we can use the above to conclude $V_n(C) \leq V^*(C)$ for any integer $n \geq 1$ and any $C \subseteq S$. $\square$

Theorem 6. *For any integer $n \geq 1$, we have that $V_n(g, C) \leq V_{n+2}(g, C) \leq V^*(g, C)$ for any guess $g \in G$ and $C \subseteq S$.*

Proof. Proposition 2 and Lemma 5 imply that inequalities from Theorem 5 also hold if we replace each LB with V, keeping subscripts the same, which was what we wanted. □

This shows we have developed an infinite system of lower-bounds for V^*. Recall how we plan to use these as stated in Theorem 1.

Theorem 7. *For any guess $g \in G$ and any $C \subseteq S$, we have $V_{2|C|+1}(g, C) = V^*(g, C)$.*

Proof. First we prove a similar statement about LB, that $LB_{2|C|+1}(C) = \text{MinTotal}(C)$ for any $C \subseteq S$. We do this by way of induction.

The base case of $|C| = 0$ is trivial. Assume then this is true for any $|C| \leq M$ for some integer $M \geq 0$, and suppose we have some C where $|C| = M + 1$. Then we have

$$LB_{2|C|+1}(C) = \min_{g \in \text{UG}(C)} |C| + \sum_{r \in R \setminus \{r^*\}} LB_{2M+1}(C_{g,r})$$

Because we are only considering $g \in \text{UG}(C)$, we have $C_{g,r} \subset C$, implying $|C_{g,r}| \leq M$. Since $2|C_{g,r}| + 1$ and $2M + 1$ are odd, Theorem 5 and the induction step imply that $LB_{2M+1}(C_{g,r}) = \text{MinTotal}(C_{g,r})$.

Using Definition 17 and Definition 13 from this completes the induction.

Now that we know $LB_{2|C|+1}(C) = \text{MinTotal}(C)$ for any $C \subseteq S$, Definition 17 and Definition 13 again can be used to achieve the desired result. □

Theorem 8. *Suppose we have an upper bound $\text{UB}(C)$ for $\text{MinTotal}(C)$. If for all $g \in G$ there exists an i_g such that $V_{i_g}(g, C) \geq \text{UB}(C)$, then $\text{UB}(C) = \text{MinTotal}(C)$*

Proof. By Theorem 4, we may 'round up' any odd i_g to an even $i_g + 1$ and would still have $V_{i_g+1}(g, C) \geq \text{UB}(C)$. Hence w.l.o.g we may assume that all i_g are even. Define $I = \max_{g \in G} i_g$, which must also be even. Theorem 5 lets us state that $V_I(g, C) \geq V_{i_g}(g, C)$ for all $g \in G$, implying that

$$\text{UB}(C) \leq \min_{g \in G} V_I(g, C) = LB_{I+2}(C) \leq \text{MinTotal}(C)$$

The construction of $\text{UB}(C)$ implies the desired result. □

These theorems are the basis for how we can determine an optimal starting guess and subsequently determine if a strategy is **provably** optimal:

1. Given a guessing game with secret set S, start with an upper bound $\text{UB}(S)$, found by Definition 11. Recall that by Theorem 1, we can use any lower bound for V^* in conjunction with $\text{UB}(S)$ to rule out which guesses could not be the starting guess of an optimal strategy. Theorem 6 gives us this lower bound for V^*.

2. Rule out any guess g for which $V_1(g, S) > \text{UB}(S)$, then rule out any for which $V_2(g, S) > \text{UB}(S)$, and so on. Do this until there is one guess left, or $V_n(g, S_W) \geq \text{UB}(S)$ for all the remaining guesses (in which case we have multiple optimal starting guesses, as shown by Theorem 8), or up until calculating $V_{2|S|+1}$ (by Theorem 7).

i	After filtering by V_i	$\min_{g \in G} V_i(g)$
1	12453	6829
2	1711	7664
3	324	7795
4	138	7826
5	1	7919
6	1	7920

Table 4: Using $V_1, \ldots, V_5$ to filter potential starting Wordle guesses, starting with 12972 possible guesses.

5 Application

We demonstrate using the general method and theorems of Section 4 on Wordle to test if our best Wordle strategy found in Section 3.2 by Definition 11 is optimal.

In Table 2, the best TOTAL for Wordle we found was 7920, so $\text{MinTotal}(S_W) \leq 7920$ where S_W is the secret set of Wordle. We then applied the method summarized at the end of the previous section to determine the optimal starting word, as well as the true value of $\text{MinTotal}(S_W)$.

We present our results in Table 4. After applying V_5, only one guess remained, SALET. This agrees with our strategy found in Section 3, in which the starting guess was indeed SALET. Although $V_5(\text{SALET}, S_W) = 7919$, one further iteration showed that $V_6(\text{SALET}, S_W) = 7920$, so not only did we find the optimal starting word, but the strategy found in Section 3 is a *provably* optimal strategy for minimizing the TOTAL.

This process of filtering by V_i to determine MinTotal was repeated for FFXIVrdle and Mininerdle, all having shown that the strategy found was optimal.

6 Conclusion

This paper produced two main contributions. First, we used combined valuations to leverage information in determining good strategies for guessing games. Second, we presented several theorems that led to a general theory for mathematically proving a certain strategy optimal, thereby avoiding a complete and exhaustive search. As stated in the introduction, the concrete results produced in this paper were focused on Wordle, but the theory and methodology apply to any game that fits the definition of a general guessing game.

We further hope that these theorems can help in applications of guessing games [Focardi and Luccio, 2012] as well as add mathematical rigour to studying optimal context representations in the field of meta-reasoning [Filman *et al.*, 1983]. Our results could also assist with developing predicates for practical guessing games [Thomason *et al.*, 2016], or possibly help an AI to learn such predicates. Our theorems could be adapted to enable an AI to determine, out of a set of possible predicates, which are the most 'discriminatory'.

In terms of future work, we would like to see this work expanded to guessing games in much looser restrictions, for example in situations where the answering function is non-deterministic. We would also like to find estimating functions that converge to the true answer in fewer iterations, and are faster to compute.

Acknowledgments

References

[Bardle, 2022] Bardle. A Shakespearian guessing game. https://bardle.newshakespeare.org/, 2022. [Online; accessed: 2023-01-11].

[Berghman *et al.*, 2009] Lotte Berghman, Dries Goossens, and Roel Leus. Efficient solutions for Mastermind using genetic algorithms. *Computers & Operations Research*, 36(6):1880–1885, 2009.

[Bestavros and Belal, 1986] Azer Bestavros and Ahmed Belal. Mastermind a game of diagnosis strategies. In *Bulletin of the Faculty of Engineering*. Alexandria University, Egypt, 1986.

[Chessle, 2022] Chessle. https://jackli.gg/chessle/, 2022. [Online; accessed: 2023-01-11].

[Doerr *et al.*, 2016] Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. Playing Mastermind with many colors. *Journal of the ACM*, 63(5):1–23, 2016.

[Dordle, 2022] Dordle. A double version of the Wordle game. https://dordlegame.io/, 2022. [Online; accessed 2022-11-15].

[FFXIV, 2022] FFXIV. FFXIV themed word game. https://ffxivrdle.com/, 2022. [Online; accessed: 2023-01-11].

[Filman *et al.*, 1983] Robert E Filman, John Lamping, and Fanya S Montalvo. Metalanguage and Metareasoning. In *Proceedings of the 8th IJCAI*, pages 365–369, Karlsruhe, Germany, 1983.

[Focardi and Luccio, 2012] Riccardo Focardi and Flaminia L Luccio. Guessing bank PINs by winning a Mastermind game. *Theory of Computing Systems*, 50(1):52–71, 2012.

[Genesereth and Björnsson, 2013] Michael Genesereth and Yngvi Björnsson. The international general game playing competition. *AI Magazine*, 34(2):107–111, 2013.

[Genesereth and Thielscher, 2014] Michael Genesereth and Michael Thielscher. *General Game Playing*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2014.

[Glazik *et al.*, 2021] Christian Glazik, Gerold Jäger, Jan Schiemann, and Anand Srivastav. Bounds for the Static Permutation Mastermind game. *Discrete Mathematics*, 344(3):112253, 2021.

[Heardle, 2022] Heardle. https://heardle.org/, 2022. [Online; accessed: 2023-01-11].

[Knuth, 1977] Donald E. Knuth. The computer as Mastermind. *Journal of Recreational Mathematics*, 9(1), 1977.

[Kooi, 2005] Barteld Kooi. Yet another Mastermind strategy. *ICGA Journal*, 28(1):13–20, 2005.

[Koyama and Lai, 1991] Kenji Koyama and Tony Lai. An interactive knowledge transfer model and analysis of "Mastermind" game. In *Proceedings of the 2nd Int. Workshop on Algorithmic Learning Theory*, pages 196–206, Tokyo, Japan, 1991.

[Koyama and Lai, 1994] Kenji Koyama and Tony Lai. An optimal Mastermind strategy. *Journal of Recreational Mathematics*, 25(4):251–256, 1994. As cited by Ville [2013].

[Serrels and Boom, 2022] Mark Serrels and Daniel Van Boom. Wordle: Everything You Need to Know About 2022's Biggest Word Game. https://www.cnet.com/culture/internet/wordle-everything-you-need-to-know-about-2022s-biggest-word-game/, Nov 2022. [Online; accessed 2023-05-11].

[Sexaginta-quattuordle, 2022] Sexaginta-quattuordle. 64 simultaneous Wordle games. https://64ordle.au/, 2022. [Online; accessed 2022-11-15].

[Sterling and Shapiro, 1994] L.S. Sterling and E.Y. Shapiro. *The Art of Prolog, second edition: Advanced Programming Techniques*. Logic Programming. MIT Press, 1994.

[Stuckman and Zhang, 2005] Jeff Stuckman and Guo-Qiang Zhang. Mastermind is NP-complete. *arXiv preprint*, arXiv:cs/0512049, December 2005.

[Thomason *et al.*, 2016] Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J Mooney. Learning multi-modal grounded linguistic semantics by playing "I Spy". In *Proceedings of the 25th IJCAI*, pages 3477–3483, New York, USA, 2016.

[Tridle, 2022] Tridle. A triple Wordle game. https://engaging-data.com/tridle/, 2022. [Online; accessed 2022-11-15].

[Ville, 2013] Geoffroy Ville. An optimal Mastermind (4,7) strategy and more results in the expected case. *arXiv preprint*, arXiv:1305.1010, March 2013.

[Wardle, 2021] Josh Wardle. Wordle - The New York Times. https://www.nytimes.com/games/wordle/index.html, 2021. [Online; accessed 2022-03-22].