Entropy as a Measure of Puzzle Difficulty

Eugene You Chen Chen[1], **Adam White**[1,2], **Nathan R. Sturtevant**[1,2]

[1] Department of Computing Science, University of Alberta
[2] Alberta Machine Intelligence Institute (Amii)
eugene@ideaowl.com, {amw8, nathanst}@ualberta.ca

Abstract

Evaluating and ranking the difficulty and enjoyment of puzzles is important in game design. Typically, such rankings are constructed manually for each specific game, which can be time consuming, subject to designer bias, and requires extensive play testing. An approach to ranking that generalizes across multiple puzzle games is even more challenging because of their variation in factors like rules and goals. This paper introduces two general approaches to compute puzzle entropy, and uses them to evaluate puzzles that players enjoy. The resulting uncertainty score is equivalent to the number of bits of data necessary to communicate the solution of a puzzle to a player of a given skill level. We apply our new approaches to puzzles from the 2016 game, *The Witness*. The computed entropy scores largely reproduce the order of a set of puzzles that introduce a new mechanic in the game. The scores are also positively correlated with the user ratings of user-created Witness puzzles, providing evidence that our approach captures notions of puzzle difficulty and enjoyment. Our approach is designed to exploit game-specific knowledge in a general way and thus can extended to provide automatic rankings or curricula in a variety of applications.

Introduction

Puzzle games vary in their rules, limitations, and goals, resulting in game-specific approaches to determine puzzle difficulty. Difficulty is also different for players of different knowledge and skill levels, further complicating measurement. However, such a measure can be very useful. It can be used to predict whether puzzles will be interesting to players of different skill levels, identify specific skills that render puzzles as easy or challenging, and help explain or generate the ordering of puzzles within puzzle games. A difficulty measure can also help develop a *curriculum* of puzzles that requires players to gain knowledge for further progression. A general measure related to difficulty can apply to educational games, or even learning in general, where the primary goal is with increasing student knowledge: adjusting problem difficulty can similarly influence levels of enjoyment and confidence, leading to further engagement and increased learning.

Several simpler puzzle measures can be generally extracted and used as proxies for difficulty. For example, the number of solutions to a puzzle provides a natural measure of difficulty because more solutions implies easier puzzles. Shorter solution lengths also relate to difficulty for similar reasons. However, these measures are not sufficient for capturing difficulty in puzzles, as we will later show.

Specific algorithms have been proposed to measure puzzle difficulty for some games. Sudoku puzzles have been measured for difficulty by mapping them into sets of constraint satisfaction problems (Ercsey-Ravasz and Toroczkai 2012). In Sokoban difficulty has been measured by the minimum number of steps needed to decompose a puzzle into subproblems (Jarušek and Pelánek 2010). While effective, these algorithms are tied to the puzzle domains they are created to measure. We aim to design general algorithms that apply across multiple puzzle games.

In this paper we contribute a new measure of puzzle difficulty based on entropy. Applied to single-player, turned-based puzzles, our entropy measure quantifies the communication that would be needed to describe a puzzle's solution, given player knowledge of the puzzle domain. Entropy is calculated from the number of choices a player faces at each step in the solution path. Using these notions, we derive two information entropy measures and then evaluate them on several puzzle test sets

Empirically, we find that our best approach, ReMUSE, is able to measure puzzle difficulty effectively, according to two experiment outcomes. First, in *The Witness*, it orders a set of mechanic-introducing puzzles very similarly to the original puzzle designers. Second, we find a positive correlation between the ReMUSE scores and user ratings for a series of online puzzles frequented by puzzle enthusiasts, indicating a possible relation to puzzle enjoyment and difficulty. The correlation with user ratings for ReMUSE are higher than the correlation with other puzzle measures such as average solution length and a puzzle's number of solutions.

Related Works

Generative approaches need a way to select intriguing puzzles, and the work in this paper is related to evaluating puzzles based on specific properties that indicate their interesting nature. Grammatical evolution can be used to create levels for a clone of a puzzle game *Cut the Rope*, measuring generated levels for playability and variety (Shaker et al.

2013). Incremental Exhaustive Content Generation (EPCG) (Sturtevant and Ota 2018) can be used to produce minor variations in the design of *Snakebird* game levels to significantly increase the length of the shortest possible solution (Sturtevant et al. 2020). In *The Witness*, neural-guided tree search has been used to determine a set of ordered puzzles (Lelis et al. 2022) that compares favourably with ordered puzzles from the game in teaching players to solve additional test puzzles. In each of these works, problem-specific scores are designed to measure and select the most interesting content.

Difficulty in puzzles has been measured using different approaches. Search and strategic depth scores have been suggested as reasonable indicators of puzzle difficulty (El-Nasr, Drachen, and Canossa 2013). Others have proposed constraint satisfaction solvers as part of a deductive search approach to measure difficulty (Browne 2013).

Difficulty and challenge are valuable properties to extract from puzzles because they can affect player enjoyment (Abuhamdeh and Csikszentmihalyi 2012): online games of Chess played against superior opponents were found to be more enjoyable than games against inferior opponents. This reflects the concept of arriving at a positive experience, so-called *flow* (Csikszentmihalyi 1990), which is arrived at when performing slightly challenging tasks, compared to tasks that are too easy or too difficult, leading to boredom or frustration respectively.

Information entropy can be represented as the uncertainty of outcomes of a random variable (Shannon 1948), and prior work has explored the connections between entropy and the perceived depth of a game, or how it enables sustained, long-term learning by players that provide dedicated problem-solving attention (Lantz et al. 2017).

Summarizing related work, measures of puzzle difficulty are used in many different applications, and there is not yet a single measure which is broadly applicable.

Background

We design two algorithms, MUSE and ReMUSE, which take as input a single-player, turn-based puzzle and return a scalar score for the entropy of a puzzle. In the upcoming subsections, we describe how we represent puzzles as graphs, utilizing simple puzzles from the game *The Witness* to illustrate essential concepts. We then describe information entropy, KL divergence, and softmin algorithms, which are utilized in MUSE and ReMUSE.

Puzzle Representation as a Graph

A single-player, turn-based puzzle can be defined as a directed graph $G = (V, E)$. V is a set of k vertices, also interchangeably called states s, where $V = \{s_1, ..., s_k\}$, and states can be reached through edges $E = \{e_1, e_2, ...\}$. Each directed edge can be represented as a pair of states originating from state s_i to s_j, or $e_m = \{s_i, s_j\}$. Beginning with the action of picking an initial starting state s_{start} from a set of starting states S_{start}, a solution to a puzzle is an edge-connected path from a start state to a goal state.

Edges E of the puzzle are not explicitly enumerated: instead, they are calculated with an action function $A(s)$ that returns a set of edges, interchangeable with actions a, for a given state s. A puzzle also needs a goal test function $\Gamma(s)$ that returns 1 if state s is a goal state and 0 if not.

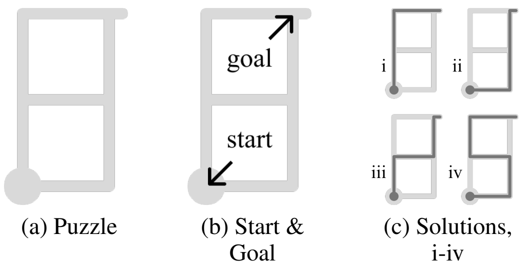


Figure 1: Example Witness Puzzle

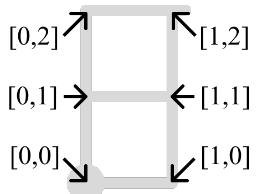
Considered together, a puzzle can be defined by a graph, a set of start states, an action function, and a goal test function. This is represented as $puzzle = \{G, S_{start}, A, \Gamma\}$. This representation is used as the input to compute our MUSE and ReMUSE scores.

The Witness Domain

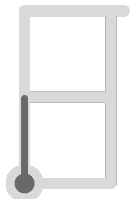
The Witness is a 2016 single-player video game where players can explore an island filled with different biomes of themed, solvable puzzles. Early puzzles are presented as $[w \times h]$ sized rectangular grids, such as the $[1 \times 2]$ sized puzzle in Figure 1a. Solving any Witness puzzle will, at minimum, require a non-crossing path to be drawn along the light-grey grid from special start junctions, indicated by round circles, to any goal junctions, indicated by notches. The start and goal junctions for our puzzle are displayed in Figure 1b, and all four solutions for the puzzle are shown in Figure 1c. Note that when a non-crossing path is drawn and reaches the end goal junction, the notch indicating the end goal junction is automatically filled as part of the drawn path.

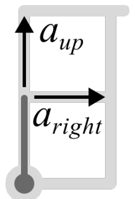
Formally, the state of a Witness puzzle is the path drawn during play, which can be represented as a sequence of junction positions. For example, junction positions for the Witness puzzle shown throughout Figure 1 and 2 are listed in Figure 2a. State s from Figure 2b is $\{[0, 0], [0, 1]\}$, starting with the only start junction in this puzzle, with $S_{start} = \{[0, 0]\}$. Graph edges or actions in Witness puzzles represent cardinal (up, down, left, right) directions used to draw a path to a next state, so action edge-detection function $A(s)$ returns $\{a_{up}, a_{right}\}$ for state s, as shown in Figure 2c. The next state will return $\{[0, 0], [0, 1], [0, 2]\}$ if a_{up} is taken at s, or $\{[0, 0], [0, 1], [1, 1]\}$ if a_{right} is taken instead. Goal test function $\Gamma(s)$ returns 0 for state s in Figure 2b, and 1 for each state shown in Figure 1c.

Unlike the puzzle in Figure 1a, most Witness puzzles also contain **constraints** that must be fulfilled when a non-crossing path is drawn from a start to goal junction. The puzzle in Figure 3a contains such a constraint: different coloured squares within the grid are associated with constraints that require that squares of only one colour exist within the regions separated by a start-to-goal non-crossing path. As a result $\Gamma(s)$ returns 1 only for states shown in iii)



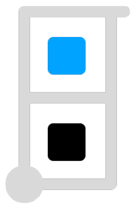
(a) Junctions and their positions

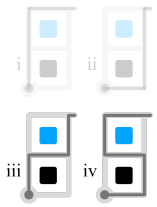

(b) Sample state s



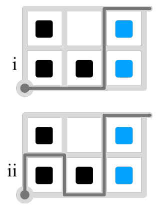
(c) Actions from $A(s)$

Figure 2: Encoding states and actions in the Witness


(a) Puzzle w constraints



(b) Solutions, iii & iv



(c) 3x2 puzzle & solutions

Figure 3: Witness Puzzles with Square Constraints

and iv) in Figure 3b since the different coloured squares are separated into two regions with the path. $\Gamma(s)$ returns 0 for states shown in i) and ii) of Figure 3b. Both states depicted in Figure 3c return 1 through $\Gamma(s)$. For i) in Figure 3c, the path separates two regions of coloured squares (3 black squares left, 2 blue squares right), while three regions (containing 1 black, 2 black, and 2 blue squares) also separate squares of different colours in ii).

Information Entropy

In information theory, entropy is a measure of the uncertainty of a random variable Z, where Z has k possible outcomes $\{z_1, ..., z_k\}$ (Shannon 1948). If the probability of an outcome z is given by a probability function $P(z)$, the **information entropy formula** is as follows:

$$H(Z) \doteq \sum_{n=1}^{k} P(z_n) \log_2 \frac{1}{P(z_n)} \tag{1}$$

For example, for an unfair double-sided coin that always turns up heads there is no uncertainty in the outcome. Mathematically, given the random variable $Z_{\text{unfair}} = \{z_{\text{heads}}, z_{\text{tails}}\}$, the probability $q = P(z_{\text{heads}}) = 1$ and $P(z_{\text{tails}}) = 0$, and so its entropy is $H(Z_{\text{unfair}}) = [1 \log_2 \frac{1}{1}] + [0 \log_2 \frac{1}{0}] = 0$ bits.

With a fair coin, where there is an equal chance that heads or tails is the outcome of the flipped coin, the probabilities $P(z_{\text{heads}}) = 0.5$ and $P(z_{\text{tails}}) = 0.5$ result in the entropy of the outcomes being $H(Z_{\text{fair}}) = [0.5 \log_2 \frac{1}{0.5}] + [0.5 \log_2 \frac{1}{0.5}] = 1$ bit. In other words, it would take 1 bit of information to communicate the outcomes of a flipped, fair coin.

KL Divergence

Kullback–Leibler (KL) Divergence (Kullback and Leibler 1951), often used in machine learning models to calculate loss (Peng et al. 2018; Sohn, Lee, and Yan 2015), is a measure that calculates the difference between two probability distributions. We use the **relative entropy** interpretation of KL divergence, or the resulting uncertainty from using a probability distribution Q to represent a true probability distribution P:

$$D_{KL}(P||Q) \doteq \sum_{x \in X} P(x) log_2 \frac{P(x)}{Q(x)} \tag{2}$$

Keeping with our example of a fair coin where the $P(Z_{fair}) = \{P(z_{\text{heads}}), P(z_{\text{tails}})\} = \{0.5, 0.5\}$, the KL divergence from using a probability distribution Q that estimates heads turning up 90% of the time is $D_{KL}(P||Q) = [0.5 \log_2 \frac{0.5}{0.9}] + [0.5 \log_2 \frac{0.5}{0.1}] = 0.74$. If Q had the same probability distribution as P, then the KL divergence would work out to be 0, reflecting no uncertainty from using Q to represent P.

Softmin Function

The softmax function (Bridle 1990) is widely used to normalize vectors of real numbers into a probability distribution on outcomes. It is, for instance, often used as the last activation function of a neural network to turn network outputs into a probability distribution for a set of output classes, and is defined as:

$$\text{Softmax}(z_i) \doteq \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{3}$$

In our work we want the highest value in the incoming distribution to have the lowest probability, so we use the similar softmin function, which uses $-z_i$ in place of z_i in the calculation.

Encoding Puzzle Uncertainty as Entropy

MUSE and ReMUSE measure the uncertainty of a puzzle, specifically encoding the uncertainty encountered in solving a puzzle. This can also be viewed as the amount of information that an *oracle*, who knows the solution to a puzzle, needs to communicate to a player so that they can make the correct decision at each step when solving a puzzle.

We utilize information entropy to encode the uncertainty encountered at each state, similar to how uncertainty is encoded for the outcomes of coins or dice. We use the number of legal actions of a puzzle at a given state s, $|A(s)|$, to determine the entropy encountered at that state.

In the Witness there are at most four cardinal actions a per state s and thus the amount of *uniform* entropy is simply:

$$H(Z_{A(s)}) = \begin{cases} 0, & \text{if } |A(s)| = 1 \\ 1, & \text{if } |A(s)| = 2 \\ 1.6, & \text{if } |A(s)| = 3 \\ 2, & \text{if } |A(s)| = 4 \\ \infty, & \text{if } |A(s)| = 0 \end{cases} \tag{4}$$

```
Algorithm 1: MUSE: Minimum Uniform Solution Entropy
 1: // Note: $\sigma$ is either $A$ or $L$, depending on approach.
 2: // If $L$, then all $\sigma(s)$ are replaced with $L(s)$.
 3: // Additionally, if an n-step lookahead is used, $L(s,n)$
    and $\Gamma(s,n)$ applies instead of $L(s)$ and $\Gamma(s)$.

 4: function MUSE($S_{\text{start}}, \sigma, n$)
 5:     $uses \leftarrow [\quad]$
 6:     for each $s_{\text{start}} \in S_{\text{start}}$ do          ▷ For every start state
 7:         Append MINPATHENT($s_{\text{start}}, \sigma, n$) to $uses$
 8:     end for
 9:     return min($uses$)
10: end function

11: function MINPATHENT($s, \sigma, n$)
12:     if $\Gamma(s) = 1$ then return 0                    ▷ Solved
13:     if $|\sigma(s)| = 0$ then return $\infty$              ▷ Unsolvable
14:     $childEnts \leftarrow [\quad]$
15:     for each $a \in \sigma(s)$ do                     ▷ For every action
16:         $s' \leftarrow$ Apply $a$ to $s$
17:         Append MINPATHENT($s', \sigma, n$) to $childEnts$
18:     end for
19:     $localEnt \leftarrow H(Z_{|\sigma(s)|})$
20:     return min($childEnts$) $+ localEnt$
21: end function
```

There are two important special cases to consider. If only one legal action is possible, then there is no uncertainty regarding which action must be taken, so entropy is 0. If, however, there are no possible actions, or $|A(s)| = 0$, then no amount of bits can encode for the solution path from state s since there are no future outcomes or successor states, and we assign ∞ for the entropy of such states.

Minimum Uniform Solution Entropy (MUSE)

Our first measure, called Minimum Uniform Solution Entropy (MUSE), computes a score for a single-player, turn-based puzzle that reflects the uncertainty encountered when the puzzle is solved. More specifically, it computes the entropy of the solution with the smallest entropy. It achieves this through a simple recursive function depicted in lines 11-21 of Algorithm 1. The entropy of any state s is the sum of the *local* entropy resulting from the number of actions $|A(s)|$ and the entropy belonging to the child state with the smallest entropy. The algorithm returns the entropy of the solution with the smallest entropy, which is particularly relevant if there is more than one start state, or $|S_{\text{start}}| > 1$.

Computed recursively, the base cases (lines 12-13) handle if the goal is reached (returning 0 entropy) or if no actions are available (returning ∞ entropy). The general case recursively computes the entropy of each child (lines 14-18) and then returns the local entropy of the current state plus the minimum entropy of the children (line 20). Note that we will later generalize the action function passed in, so σ is the action function in this pseudo-code.

Figure 4 provides a concrete example of applying Algorithm 1 to a puzzle from the Witness. State 1 from the figure is also the puzzle's start state $s_{start} = \{[0,0]\}$, which has actions a_{up} and a_{right} returned by the action function $A(s)$. These lead to States 2a and 3a, or $\{[0,0],[0,1]\}$ and $\{[0,0],[1,0]\}$ respectively, since states in Witness puzzles are the non-crossing paths that can be represented by a series of junction positions.

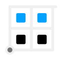
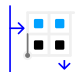
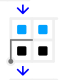
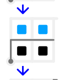
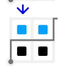
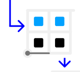
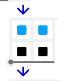
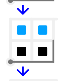
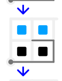
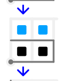
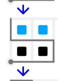
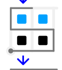
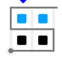
ID	State	Position	$\lvert A\rvert$	$H(Z_{\lvert A\rvert})$	$\lvert L\rvert$	$H(Z_{\lvert L\rvert})$
1		[0,0]	2	1	2	1
2a		[0,1]	2	1	1	0
2b		[1,1]	3	1.6	1	0
2c		[2,1]	2	1	1	0
2d		[2,2]	0	0	0	0
3a		[1,0]	2	1	2	1
3b		[2,0]	1	0	1	0
3c		[2,1]	2	1	1	0
3d		[1,1]	2	1	1	0
3e		[0,1]	1	0	1	0
3f		[0,2]	1	0	1	0
3g		[1,2]	1	0	1	0
3h		[2,2]	0	0	0	0

Figure 4: Entropy encountered at each state for puzzle solution reached at states 2d and 3h using valid action function return $A(s)$ and logic action function return $L(s)$.

MUSE computes the sum of the local entropy encountered at State 1 plus the minimum sum of entropy from the next state with the lowest entropy. This would be $H(Z_{|A(s)|}) = 1$ bit of entropy for the $|A(s)| = 2$ actions at State 1 plus the lower of the two entropies from State 2a and 3a, which themselves would need to be similarly calculated in a recursive fashion.

We can easily calculate the entropy at State 2a and 3a through three observations. Only two solutions exist for the puzzle, and as a result all other states not shown at Figure 4 will have an entropy of ∞, since all other states eventually end up at a terminal, non-goal state where $|A(s)| = 0$. Only goal states of 2d and 3h would return 0 entropy to parent states, and because the minimum child entropy is picked at each state, all other child entropy resulting in ∞ would be ignored. Because States 2a and 3a do not branch into multiple solutions, we sum child entropy together. Thus the entropy at State 2a is its local entropy plus the entropy of its descendant states, or $1 + 1.6 + 1 = 3.6$ bits of entropy. Similarly, the entropy at State 3a is $1 + 0 + 1 + 1 + 0 + 0 + 0 = 3$ bits of entropy.

Together, the MUSE for the puzzle is $1 + \min(3.6, 3) = 4$ bits of entropy, from reaching the goal state at 3h, or $s_{\text{goal}} = \{[0,0], [1,0], ..., [1,2], [2,2]\}$.

MUSE is a measure that computes a score for a single-player, turn-based game that reflects the uncertainty encountered while a puzzle is solved. The uncertainty is encountered by human players, but so far we have treated any action returned by $A(s)$ as a source of uncertainty, when players with skill, knowledge, and experience might not see all actions as equally valid in achieving a solution. We can improve our approach by incorporating player knowledge and skills within MUSE, and to do that, we introduce logic-driven inference rules.

Logic Puzzles and Inference Rules

Logic puzzle players typically deduce a set of rules that aid them in solving puzzles. For example, with typical rectangular jigsaw puzzles, players can intuit that puzzle pieces containing two straight edges are its corner pieces, and those corner pieces must be adjacently connected with other pieces that have a straight edge. Accurate deductions can be translated into **inference rules**, or a set of rules that must be followed for the puzzle to remain solvable.

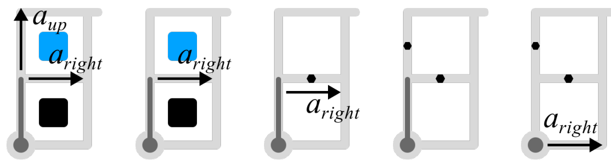
Inference rules reduce the uncertainty found in logic puzzles, and we use them to *model* the uncertainty encountered by skilled players while solving puzzles. Continuing with the example of jigsaw puzzles, experienced players know that it is unnecessary to connect pieces with a straight edge with ones that do not. This drastically reduces the number of connections players need to attempt. Inference rules reduce the number of actions experienced players need to consider, potentially reducing the uncertainty experienced at every turn of a turn-based puzzle.

While these rules can be created automatically (Nakano et al. 1998; Stevens, Bulitko, and Thue 2023), in this work we manually created sets of inference rules for Witness puzzles to generate specific entropy-related scores based on the uncertainty found in solving a puzzle.

Inference Rules for Witness Puzzles

Inference rules exist for different constraints found in Witness puzzles. For instance, while it is possible to extend the path either up or right for the next part of the path for the puzzle in Figure 5a, we can apply an inference rule that eliminates a_{up} as a potential action to take. The rule applies as follows: at a junction adjacent to two different colour squares, the action extending the path to immediately divide these squares **must** be taken. Not doing so would result in the puzzle not being solvable since the pieces have to be in separate regions, and if we do not separate them immediately, there will be no way to do so later.

This inference rule can be formalized as follows. We introduce a logic function $L(s)$ that incorporates the inference rule for immediately separating different colour squares. Consider the puzzle shown in Figure 5a with $A(s) = \{a_{\text{up}}, a_{\text{right}}\}$. In state $s = \{[0,0], [0,1]\}$ at Figure 5b, $L(s)$



(a) A(s) actions (b) L(s) action (c) L(s) action (d) L(s) actions (e) L(s,n) with n=1

Figure 5: Comparing A(s) with Logic-Infused L(s)

would return a_{right} as the only action that could lead to the goal state, or $\Gamma(s) = 1$ for a future state s.

Another inference rule can be written for *must-cross* constraints. These constraints are black hexagons (which look like black dots in our smaller diagrams) along the grid lines of Witness puzzles. When must-cross constraints are present, they must be crossed over by a path for the constraints to be fulfilled. The inference rule for must-cross constraints is simple: at a junction where a must-cross constraint lies between two junctions, the action extending the path to cross that must-cross constraint must be immediately taken. As a result, $L(s)$ returns only a_{right} for the puzzle state shown in Figure 5c.

$L(s)$ can return no actions if a combination of inference rules suggest no future states beyond s can lead to a goal state, even though $A(s)$ would return a set of actions. This is the case of the puzzle in Figure 5d, where inference rules require both a_{up} and a_{right} to reach a possible goal state. Since that is not possible from s, $L(s)$ returns no actions.

Modeling player lookahead Uncertainty can be further reduced if we account for a player's ability to consider the consequences of actions a few steps ahead in turn-based games, as good chess players typically do (Campitelli and Gobet 2004). In our case, we extend $L(s)$ to $L(s, n)$ to return a set of logic-driven actions at s that check if the puzzle is still solvable after a n-step depth first search (DFS), or is solved in the DFS, suggesting 0 entropy. Having already established that the puzzle state from Figure 5d cannot result in reaching future goal states, $L(s, 1)$ for the start state shown in Figure 5e will only return a_{right}.

MUSE with a Modified Action Function

We augment MUSE with inference rules by simply replacing valid action function A with logic action function L for σ as shown in Algorithm 1. In some cases $A(s)$ and $L(s)$ are exactly the same: in the example puzzle shown in Figure 4, no local inference rules apply to L at State 1, so $|A(s)| = |L(s)| = 1$. However, in many other states like State 2a or 2b, the inference rule pertaining to coloured squares makes a_{right} the only logical action that can be taken, $L(s) = \{a_{\text{right}}\}$, and the local entropy at that state, not accounting for child entropies, is 0.

As a result, the uncertainty encountered in solving the puzzle, or the puzzle MUSE score, is reduced. This can be seen for the solutions reached at States 2d and 3h in Figure 4. Similar to the earlier derivation of the MUSE score, we can first sum up their local and descendant entropies:

```
Algorithm 2: ReMUSE: Relative MUSE
    ...
20: $localEnt \leftarrow$
        $KL(softmin(childEntropies), (Z_{|L(s)|}))$
    ...
```

for State 2a this is $0 + 0 + 0 = 0$ and for State 3a it is $1 + 0 + 0 + 0 + 0 = 1$. Therefore, the puzzle's MUSE score is the local entropy at State 1 plus its smallest child entropy, or MUSE $= 1 + \min(0, 1) = 1$, which is smaller than the entropy of 4 computed using only the valid action function A.

ReMUSE: Relative MUSE

Our MUSE algorithm with inference rules, which we will simply refer to as our general MUSE algorithm from here, encodes the uncertainty of a puzzle experienced by a modeled player. This uncertainty is measured locally and uniformly at each state, but does not account for the difference in entropy from different actions. To account for this, we introduce our ReMUSE approach.

Several cases need to be accounted for in the design of our ReMUSE algorithm. Consider a state s where three actions $|L(s)| = 3$ exist, but the eventual outcome of all three actions is that the puzzle will be solved. MUSE returns a local entropy score of 1.6 for state s despite the intuition that in such a state, no further information needs to be encoded or communicated to the player to solve the puzzle. The entropy should be 0 in such a state. A similar situation occurs if all children have the same entropy: the choice again does not matter, and no information needs to be communicated to describe the solution. Finally, consider if the three actions have only an ϵ difference in entropy: even if a player takes the wrong action, the entropy of the rest of the puzzle is essentially unchanged.

ReMUSE utilizes KL Divergence to account for the *relative entropy* between two probability distributions when computing the local entropy of a given state. The first distribution is a uniform probability distribution based on the number of actions. The second distribution is the softmin of the entropy from the immediate children. The larger the KL divergence measure, the larger the divergence from the assumption of uniform probability.

This approach only differs from the MUSE algorithm shown in Algorithm 1 by a single line (line 19), highlighted in Algorithm 2. Instead of setting a state's local entropy as the uniform information entropy of the state, we replace it with a KL divergence comparison between the softmin of the entropy of the children and the uniform distribution.[1]

Other Puzzle Measures

Single player, turn-based puzzles from different puzzle games share a set of general measures with which we can compare our entropy-related scores from MUSE and ReMUSE. The number of solutions and the average length of solutions provide reasonable measures of puzzle difficulty. The number of solutions can be determined by the number of states that return $\Gamma(s, 0) = 1$ after a brute-force traversal of every path. The average length of solutions takes the average number of actions needed to arrive at a solution for the puzzle. We expect that for puzzles of a given $[w \times h]$ size, the more difficult puzzles are ones with fewer number of solutions and larger average length of solutions.

Experiment Setup and Results

We setup our experiments to answer a number of questions. First, can our proposed puzzle scores predict puzzle difficulty on user-submitted puzzles from a site frequented by puzzle enthusiasts? How do MUSE and ReMUSE compare with measures like solution length in capturing puzzle difficulty? We also compare our ReMUSE-based orderings of puzzles to those found within the Witness game, as well as the neural-ordered equidistant puzzles from prior work on puzzle ordering (Lelis et al. 2022).

Interesting Puzzles from the Windmill

We collected a series of Witness puzzles from an online site called the Windmill (Gruen 2016) where users submit puzzles and can upvote or downvote those puzzles (which we define as user ratings). We compute MUSE and ReMUSE scores, the number of solutions, and the average solution length for all puzzles. Our hypothesis is that they measure difficulty, and test this by seeing if the scores correlate with user ratings. We make this hypothesis with two assumptions. First, the players that solve puzzles on the Windmill have likely played, if not completed, *The Witness*, and can be generally considered advanced players. Second, advanced players positively upvote challenging puzzles, and downvote trivial ones.

We downloaded every single puzzle from the Windmill up to November 29th, 2022, and then filtered puzzles by a number of criteria. First, we only computed puzzle scores for puzzles containing one or a combination of must-cross, rounded-square, and cannot-cross (briefly described in the final paragraph of this section) constraints, since these were the constraints our puzzle solver could account for, and ones we had inference rules written into our code. We also filtered specifically for puzzles that had a $[4 \times 4]$ size. This was done for a number of reasons: a fixed grid size allows for a consistent assessment and evaluation of puzzles that have similar number of states, thereby removing the element of variable scoring due to different puzzle sizes. This grid size also limits a puzzle's maximum entropy, as we believe that players do not enjoy puzzles with entropy that is too high. We also removed exact copies of puzzles that appear in the game. Finally, we removed five outlier puzzles with user ratings of above 40: these were created and voted upon early on when the Windmill was launched.

After filtering, 104 puzzles fit our criteria for testing with an average of 6.4 upvotes. We also adjusted user ratings to reflect the smaller number of players participating on the

[1] We use a uniform prior distribution to model the probability of player actions. If a player has a different prior distribution or policy, we can use that instead.

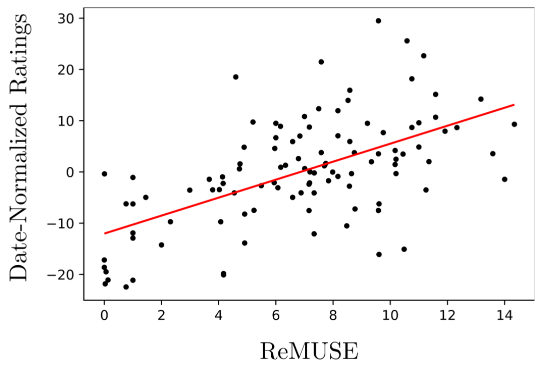


Figure 6: User ratings of puzzles and their ReMUSE scores. An interactive version of these results with all the data and each playable Witness puzzle is available at https://ideaowl.com/remuse

Approach	Correlation	p-value
MUSE, no lookahead	0.41	1.1E-05
MUSE, n=1 step lookahead	0.40	2.4E-05
MUSE, n=2 step lookahead	0.40	2.4E-05
ReMUSE, no lookahead	**0.57**	1.5E-10
ReMUSE, n=1 step lookahead	0.56	4.9E-10
ReMUSE, n=2 step lookahead	0.56	4.2E-10
Number of Solutions	0.32	9.7E-04
Average Solution Length	0.47	3.9E-07

Table 1: Correlation (Pearson Correlation Coefficients) and p-values with user ratings for Windmill puzzles

Windmill site over time by identifying the correlation between user ratings and time, creating a temporally adjusted user rating for each puzzle.

The scores obtained from applying different approaches to user-generated puzzles from the Windmill provide a number of interesting insights. As seen on Table 1, MUSE scores on puzzles correlate moderately with the puzzle's user ratings, with ReMUSE scores leaning towards a strong correlation at 0.57. ReMUSE scores for every puzzle are visualized in Figure 6. Unintuitively, it seems that performing n-step lookaheads with either MUSE or ReMUSE approaches lowers the entropy scores' correlation with user ratings. Looking at other measures, the number of solutions for puzzles had a low correlation to user rating. We expect that puzzles with fewer solutions would be more strongly correlated to higher user ratings since fewer solutions imply more difficult puzzles.

These results suggest that the uncertainty-based MUSE and ReMUSE approaches can be used to predict puzzle enjoyment, and by extension we argue that it can predict puzzle difficulty. When we see a positive correlation with user ratings, we are likely seeing a proxy of encountered difficulty since players from the Windmill are experienced and likely derive more enjoyment from challenging puzzles, suggesting that our approaches can be used to assess puzzle difficulty.

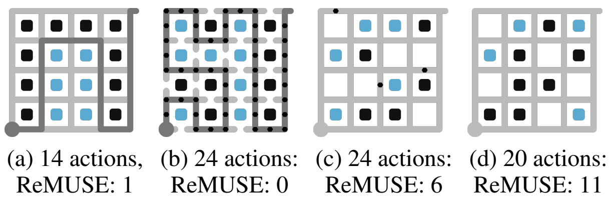

(a) 14 actions, ReMUSE: 1
(b) 24 actions: ReMUSE: 0
(c) 24 actions: ReMUSE: 6
(d) 20 actions: ReMUSE: 11

Figure 7: Puzzles on the Windmill with only 1 solution

Comparing the ReMUSE measure to the measures of the number or the length of solutions yields two additional findings. Our approaches compare well, if not notably better, in correlating with user ratings. Given that these other measures are often associated with predictors of difficulty, this is a good result. Perhaps more meaningful, however, is the discovery that our puzzle scores can vary greatly even when those other measures are held constant. For example, puzzles from Figure 7 have only one single solution but vary in the number of actions necessary to solve them, as well as their ReMUSE scores. Puzzles a) and b) are solved, noting that for b) the gaps/breaks in the grid mean that no path can be drawn across the gap: these are also known as *must-not cross constraints*. Puzzle a) is simple, but b) is comparatively trivial to solve despite requiring more actions to solve it. Like Puzzle b), Puzzle c) also requires 24 actions to reach the only solution for the puzzle, but is more challenging than b) and has a ReMUSE score of 6. Puzzle d) has fewer actions necessary than b) or c), but has the highest ReMUSE score of the four puzzles, suggesting a higher level of difficulty. For reference, puzzles a) to d) have respective user rating scores of -6, -15, 15, and 28.

Patterns from Ordered Witness Puzzle Sets

We consider puzzles from two sets of Witness puzzles, with the first being the starting slate of puzzles within *The Witness* that introduce constraints to the player. We start by computing ReMUSE scores for these puzzles. Sorting puzzles by the ReMUSE scores, we can look for patterns that can help to explain the ordering of puzzles with the game.

We also evaluate Witness puzzles and compare their ordering with the set of *equidistant* puzzles from work on learning curriculum (Lelis et al. 2022). Given that the equidistant curriculum was generated and shown as effective in helping players learn to solve later puzzles, a similar ordering of the puzzles can provide a possible relationship between of the perceived entropy of a puzzle and its difficulty as modeled through their approach.

Comparing the order of the introductory constraint puzzles from the Witness game with the ordering from ReMUSE provides a number of insights. First, as seen in Figure 8, the overall order is very similar. ReMUSE scores the first three puzzles as trivial and equivalent in uncertainty, while puzzles vi-ix have the exact same order that the ReMUSE scores for. Only one change in order occurs as a swap for puzzle iv and v. Both have the same MUSE score of 1 bit of entropy that occurs at state $s_{\text{start}} = \{[0, 0]\}$, but the KL

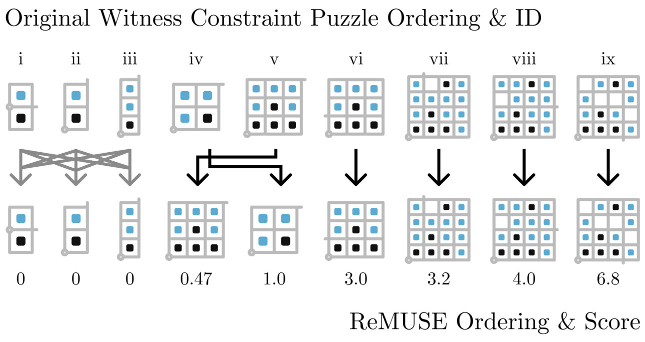


Figure 8: Puzzle order of introductory constraint puzzles in *The Witness* compared with ReMUSE score-driven order

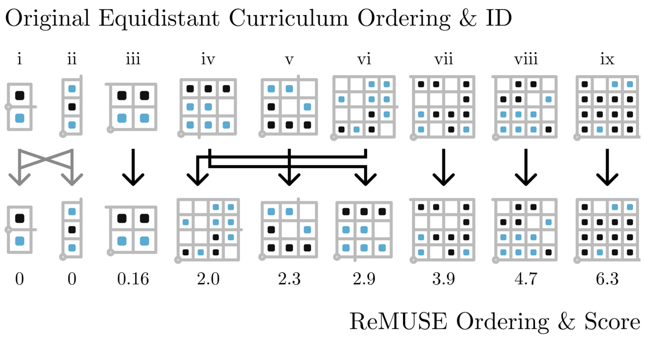


Figure 9: Puzzle order of equidistant puzzles compared with ReMUSE score-driven order

divergence of the softmin in the ReMUSE measure accounts for the two possible solutions (by first taking either a_{up} or a_{right}), so even if the shorter solution is missed by taking a_{up} the player can still solve the puzzle, resulting in lower overall entropy at s_{start} with ReMUSE. Note that we assume the player knows the constraints, but they do not when these puzzles are first introduced to them. This may account for the difference in the ordering of earlier puzzles.

The similar ordering is a promising result, as it suggests that ReMUSE can be used to create sets of increasingly challenging puzzles to teach concepts. This is what happens in *The Witness*: the game designer creates a variety of puzzles and an ordering of the puzzles to allow players to learn inference rules about the game. This ordering of increasingly challenging puzzles should not deter the player from continued play, either because the puzzles are too simple or too difficult. As demonstrated here, the ReMUSE approach is able to credibly rank puzzles for difficulty.

The ordering of puzzles through our ReMUSE measure can also help to explain the choice of puzzle ordering by a game designer. Puzzles iv and vi, as well as vii through ix, are similar to one another: in all five puzzles, four black square constraints sit beside the start junction of the puzzle. What mostly changes is the position of the exit junction, and it is done to force the player to evaluate another, more uncertain path to solve the puzzle. For example, instead of taking path of least uncertainty to a solution by first taking a_{up} at s_{start} for puzzle v), puzzle vi) removes the option for taking that path, requiring the player to encounter uncertainty for the first three states by taking the solution path starting with a_{right}.

The similar ordering for the equidistant curriculum puzzles in Figure 9 further reinforces the validity of using ReMUSE as a measure of difficulty and for ordering puzzles for learning. A neural-guided tree search modeled the difficulty of a set of generated puzzles, and 9 were selected for their equally increasing gaps in difficulty. They were successfully tested as possible replacement puzzles for the introductory constraint puzzles in the Witness game, and implies that ReMUSE would have done well at ordering another set of puzzles to teach concepts and logic to players.

These puzzles also suggest insights about entropy in different size puzzles, work that we explore further by exhaustively generating Witness puzzles of different grid sizes and comparing their ReMUSE score distributions in an upcoming theses (Chen 2023). First, ReMUSE scores vary across different size puzzles, and as shown in both sets of ordered Witness puzzles, smaller puzzles can have larger ReMUSE scores than their larger counterparts. This is promising as it suggests that ReMUSE can identify more challenging puzzles that have a smaller size rather than relying on grid size to determine difficulty. Additionally, 0-entropy puzzles exist on any puzzle size, and the larger the puzzle the longer the maximum path, and the greater the maximum entropy.

Slitherlink Puzzles and Triangle Constraints

As a separate, independent experiment, we also computed ReMUSE scores, the number and average solution length of 152 Witness puzzles from the Windmill that contain only triangle constraints. These puzzles are similar to puzzles from the game *Slitherlink*, also known as Fences or Takegaki. A full analysis is beyond the scope of this paper, but the preliminary results suggest that ReMUSE can be used to measure difficulty for *Slitherlink* puzzles (Chen 2023). Compared to the Windmill puzzles discussed earlier in this paper, ReMUSE scores for puzzles containing only triangle constraints had a lower absolute correlation to user ratings, but the relative positive difference to all other measures were much higher.

Limitations and Future Work

Inference rules are hard to write and discover. A potential solution to this problem is to use inductive logic programming (Muggleton and De Raedt 1994) to automatically create sets of inference rules by learning from databases of puzzles and their solutions (Stevens, Bulitko, and Thue 2023). These automatically generated inference rules could then be applied to our entropy algorithms.

People also vary in knowledge and skill, so certain players, like beginners, may find certain puzzles difficult because they have not yet developed their own internal set of inference rules. Our experiments model the strong player that does not make mistakes based on the inference rules we know, which works well for trying to predict the difficulty of a puzzle for experienced players. A natural next step is to create models of players with different skillsets and knowledge: they may not know all the inference rules and get stuck at a puzzle unless that gap in their knowledge is filled.

Finally, the approach to evaluating uncertainty can be extended to non-game settings. Injecting measurable uncertainty into educational settings could help learners better absorb lessons by providing skill-appropriate questions that are not too challenging or simple.

Conclusion

In this paper we introduced MUSE and ReMUSE as measures of puzzle difficulty based on entropy. We imbued our algorithms with logical inference rules to better reflect how people solve puzzles. We evaluated our approach on Witness puzzles, comparing our entropy scores with user ratings from a large online database. Our results suggest that our ReMUSE measure produces scores that correlate well with user ratings and better than the correlations from alternative measures like solution length and the number of solutions. Finally, our approach produced similar orderings for puzzles in two sets of puzzle curricula, and had a positive correlation to user ratings for Witness puzzles that are a close equivalence to a different puzzle game.

Acknowledgements

This work was supported by the National Science and Engineering Research Council of Canada Discovery Grant Program and the Canada CIFAR AI Chairs Program.

References

Abuhamdeh, S.; and Csikszentmihalyi, M. 2012. The Importance of Challenge for the Enjoyment of Intrinsically Motivated, Goal-Directed Activities. *Personality and Social Psychology Bulletin*, 38(3): 317–330.

Bridle, J. S. 1990. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications*, 227–236. Springer.

Browne, C. 2013. Deductive search for logic puzzles. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, 1–8.

Campitelli, G.; and Gobet, F. 2004. Adaptive expert decision making: Skilled chess players search more and deeper. *ICGA Journal*, 27(4): 209–216.

Chen, E. 2023. Entropy as a Measure of Puzzle Difficulty. Forthcoming.

Csikszentmihalyi, M. 1990. *Flow: The Psychology of Optimal Experience*. Harper & Row. ISBN 0060162538.

El-Nasr, M. S.; Drachen, A.; and Canossa, A. 2013. *Game Analytics: Maximizing the Value of Player Data*. Springer Publishing Company, Incorporated. ISBN 1447147685.

Ercsey-Ravasz, M.; and Toroczkai, Z. 2012. The chaos within Sudoku. *Scientific reports*, 2(1): 725.

Gruen, M. 2016. The Windmill. https://windmill.thefifthmatt.com. Accessed: 2023-05-26.

Jarušek, P.; and Pelánek, R. 2010. Difficulty rating of sokoban puzzle. In *STAIRS 2010*, 140–150. IOS Press.

Kullback, S.; and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86.

Lantz, F.; Isaksen, A.; Jaffe, A.; Nealen, A.; and Togelius, J. 2017. Depth in Strategic Games. In *AAAI Workshops*.

Lelis, L. H.; Nova, J. G.; Chen, E.; Sturtevant, N. R.; Epp, C. D.; and Bowling, M. 2022. Learning Curricula for Humans: An Empirical Study with Puzzles from The Witness. In *IJCAI*, 3877–3883.

Muggleton, S.; and De Raedt, L. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19: 629–679.

Nakano, T.; Inuzuka, N.; Seki, H.; and Itoh, H. 1998. Inducing shogi heuristics using inductive logic programming. In Page, D., ed., *Inductive Logic Programming*, 155–164. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-69059-7.

Peng, X. B.; Kanazawa, A.; Toyer, S.; Abbeel, P.; and Levine, S. 2018. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*.

Shaker, M.; Sarhan, M. H.; Naameh, O. A.; Shaker, N.; and Togelius, J. 2013. Automatic generation and analysis of physics-based puzzle games. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, 1–8.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3): 379–423.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28.

Stevens, J.; Bulitko, V.; and Thue, D. 2023. Solving Witness-type Triangle Puzzles Faster with an Automatically Learned Human-Explainable Predicate. *arXiv preprint arXiv:2308.02666*.

Sturtevant, N.; Decroocq, N.; Tripodi, A.; and Guzdial, M. 2020. The Unexpected Consequence of Incremental Design Changes. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1): 130–136.

Sturtevant, N.; and Ota, M. 2018. Exhaustive and semi-exhaustive procedural content generation. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 14, 109–115.