Generalized Entropy and Solution Information for Measuring Puzzle Difficulty

Junwen Shen[1], **Nathan R. Sturtevant**[1,2]

[1]University of Alberta, Department of Computing Science
[2]Alberta Machine Intelligence Institute (Amii)
{junwen5, nathanst}@ualberta.ca

Abstract

Metrics for problem difficulty are used by many puzzle generation algorithms, as well as by adaptive algorithms that want to provide players with the puzzles at the correct level of difficulty. A recently proposed general metric, puzzle entropy, combines an analysis of game mechanics with a model of player knowledge in the form of inference rules to predict problem difficulty. The entropy of a puzzle is the amount of information required, given a player's knowledge about the puzzle, to describe a solution to a puzzle. This paper generalizes the concepts of puzzle entropy and solution information, providing a better foundation for the previous work and creating new algorithms, Minimum Solution Information and Total Solution Information. While functionally similar to past work, the new algorithms allow knowledge about a puzzle to be represented as a policy. We then evaluate the impact of inference rules, policies, and player knowledge in the 2016 game, *The Witness*.

Introduction

Developing a universal algorithm to measure puzzle difficulty is a complex task (van Kreveld, Loffler, and Mutser 2015). Different games possess various features that may be challenging to compute and standardize. For example, for *Sokoban*-type games like *Fling!* (2011), simple metrics such as the number of reachable states or problem decomposition using subproblem solution length have proven effective (Sturtevant 2021; Jarušek and Pelánek 2010). However, these metrics may not be applicable for puzzles in *Cut The Rope* (2010) and *Angry Birds* (2009), which involve real-time physics. Additionally, difficulty varies among players with different levels of knowledge and skills. Nevertheless, an accurate difficulty metric can be highly beneficial. Puzzle designers could use it to determine the quality and suitability of a puzzle within a game (De Kegel and Haahr 2020) and predict player enjoyment. It could also aid in generating and analyzing puzzle curricula, allowing players to acquire game-specific knowledge more easily (Lelis et al. 2022).

Recent work by Chen, White, and Sturtevant (2023) introduced the notion of puzzle difficulty based on the amount of uncertainty a player might face, which can be broadly applied across different puzzle games (del Solar-Zavala, Schaa, and Barriga 2023). Their approaches rely on information entropy, representing the amount of communication required to describe the solutions of a single-player turn-based puzzle. Published results demonstrated a positive correlation between puzzle difficulty, as measured by their algorithms, and user ratings on a subset of puzzles and constraints from *The Witness* (2016). This paper studies the nature of the underlying algorithms, considers enhanced player models, and expands the experiments conducted in previous studies.

This paper establishes the mathematical connection between the previous MUSE approach and the probability of solving a puzzle. New approaches are derived which have stronger mathematical foundations, while maintaining generality. This enables us to evaluate both policies and player inference rules. We then develop additional inference rules that can be applied to a broader class of puzzles in the game *The Witness*. Consequently, we are able to expand the puzzle datasets used for evaluation. Empirically, we find that our approach, total solution information (TSI), is as effective as previous methods across different datasets. We hypothesize that using too many or too powerful inference rules overestimates players' abilities, resulting in a poorer correlation between puzzle difficulty and engagement. Moreover, we evaluate a machine learning (ML) policy and show that it has learned puzzle difficulty, but not as well as our inference rules. Finally, we create a puzzle-design tool with a web GUI, enabling designers to invoke exhaustive procedural content generation (EPCG) queries that generate suggested designs sorted by difficulty from our approach.

Background and Related Works

The work in this paper is closely related to measuring puzzle difficulty and its relationship with player engagement. Puzzle difficulty represents the challenge that players will face, and it affects player enjoyment (Abuhamdeh and Csíkszentmihályi 2012). Furthermore, automated difficulty evaluation accelerates the puzzle generation pipeline and helps user-specific puzzle generation (van Kreveld, Loffler, and Mutser 2015). Simple metrics like minimum solution length are used in exhaustive procedural content generation (EPCG) (Sturtevant and Ota 2018) for incrementally designing levels in *Snakebird* (2015) (Sturtevant et al. 2020) or sliding puzzles like *Rush Hour* (De Kegel and Haahr 2020). Constraint-based algorithms, such as answer set program-

ming, were used to assess maze-like puzzles, where mazes are represented as grid-embedded trees with constraints on reachability and path length (Smith et al. 2012; Nelson and Smith 2016). More complex methods, like simulating playtests using deep reinforcement learning, can be used in *Angry Birds* (2009) (Roohi et al. 2021). Chen, White, and Sturtevant (2023) proposed an algorithm that uses heuristic search and information entropy for turn-based puzzle games, evaluated on puzzles in *The Witness* (2016).

Heuristic Search

A puzzle P can be formally defined as $P = (S, A, T, s_0, G)$ (Lelis et al. 2022), where S is a set of states that includes the initial state s_0, and G is a subset of S that represents the set of goal states. The action function A returns a set of possible actions at a given state s. If a state s is terminal, $A(s) = \varnothing$. The deterministic transition function T takes a state s and an action a as input, and produces a new state s' which is the result of applying action a to state s. Thus, the solution path to goal state g can be represented as a sequence of state-action pairs $\tau_g = \{(s_i, a_i) \mid T(s_i, a_i) = s_{i+1}$ and $T(s_m, a_m) = g \in G\}_{i=0}^{m}$, and a player's policy can be represented by a function $\pi(a, s) : A(S) \times S \to [0, 1]$ that returns the probability of taking action a at state s. The search space is assumed to be a tree, where each node corresponds to a state in S, and the tree is rooted at s_0. A directed edge exists from state s to state s' if $T(s, a) = s'$ for some action $a \in A(s)$. The successor function, $\sigma(s) = T(s, A(s))$, returns the set of successor states of s. Therefore, the entire search state space is described by the tree of all possible actions and a solution path is a path starting from initial state s_0 to a goal state g.

Entropy of a Puzzle

Information entropy (Shannon 1948) serves as a fundamental metric in information theory, allowing for the quantification of uncertainty in stochastic processes. Entropy represents the expected amount of information content given a discrete random variable X and its probability distribution $P : \mathcal{X} \to [0, 1]$ over the sample space $\mathcal{X}$.

$$H(X) = \mathbb{E}\left[I(X)\right] = -\sum_{x \in \mathcal{X}} P(x) \log_2 P(x) \qquad (1)$$

For example, consider the case of tossing a fair six-sided die. The entropy of this event would be $H = -6 \times \frac{1}{6} \log_2 \frac{1}{6} = \log_2 6 \approx 2.585$ Sh[1], indicating that the information content of each outcome is $\log_2 6$ Sh.

To gauge puzzle difficulty, we adopt the minimum uniform entropy (MUSE) and relative minimum uniform entropy (ReMUSE) metrics proposed by Chen, White, and Sturtevant (2023). These metrics quantify the amount of information an oracle would need to provide for a player to solve the puzzle. MUSE directly calculates the minimum amount of uniform entropy based on the number of choices at each step along the solution path. On the other hand, ReMUSE utilizes the Kullback-Leibler (KL) Divergence (Kullback and Leibler 1951), also known as relative entropy, as shown in Equation 2, to capture the information of other solutions from successors of the current state.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log_2 \frac{P(x)}{Q(x)} \qquad (2)$$

Here, P and Q are discrete probability distributions defined on the same sample space $\mathcal{X}$. In ReMUSE, P is the softmin of direct successors' uniform entropy, Q is a uniform distribution as player's policy, and $|P| = |Q| = |A(s)|$. MUSE and ReMUSE values increase with more complex puzzles, as they require the player to make more decisions. Previous work (Chen, White, and Sturtevant 2023) demonstrated that MUSE and ReMUSE can effectively assess puzzle difficulty. The positive correlation between these metrics and user ratings in the puzzle game, *The Witness*, further supports the relationship between difficulty and player satisfaction.

Puzzles from The Witness

The Witness is a single-player puzzle solving game published by Thekla, Inc. in 2016. We refer to them henceforth as "Witness puzzles". Most puzzles in the game are rectangular grids of size $w \times h$. In Witness puzzles, the player must draw a non-self-crossing path along the grid lines starting from the designated start junction and ending at any of the goal junctions. The start junction is represented by a round circle, and the end junction is indicated by a notch extended from the grid line (Figure 1a). When the path reaches the end junction, the notch is automatically filled.

The game includes two classes of constraints: path constraints and region constraints. Path constraints are placed on the grid lines, while region constraints are present in the blocks surrounded by the grid. Figure 1b and Figure 1c illustrate examples of path constraints and region constraints with their corresponding solutions. Additionally, Figure 2a presents a more complex puzzle that includes every type of constraint studied in this research. The requirements for satisfying these constraints are explained in Table 1. It is essential for the solution path to comply with all the constraints in the puzzle. Moreover, some combinations of constraints significantly reduce the number of possible solutions. By understanding these constraints, players can reduce the number of possible actions, thereby simplifying the puzzle. These guidelines are known as inference rules. We explore the effects of inference rules in later sections.

The solution path for the puzzle in Figure 1b is $\{(s_0, a_{\text{up}}), (s_1, a_{\text{right}}), (s_2, a_{\text{up}})\}$, and its MUSE value without using inference rules is $\sum_{i=0}^{2} \log_2 |A(s_i)| = 1{+}1{+}1 = 3$ Sh. Similarly, the solution path for the puzzle in Figure 1c is $\{(s_0, a_{\text{right}}), (s_1, a_{\text{up}}), (s_2, a_{\text{left}}), (s_3, a_{\text{up}}), (s_4, a_{\text{right}})\}$, and its MUSE value is 1 Sh instead. Based on these calculations, we conclude that puzzle in Figure 1c is easier than the one in Figure 1b.

Mathematical Foundations

We study the nature of MUSE by examining the connection to the probability of finding a solution given a player policy.

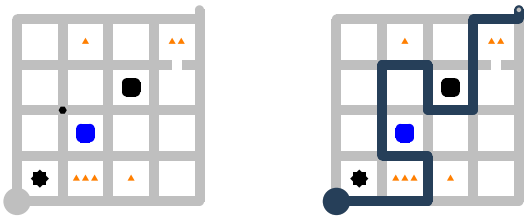
[1]We use the *Shannon* (binary unit) as the unit of information instead of *bit* (binary digit) according to the International System of Quantities (International Organization for Standardization 2008)

(a) A 1×2 puzzle with no constraint

(b) A 1×2 puzzle with 2 region constraints and a solution

(c) A 1×2 puzzle with a path constraint and a solution

Figure 1: Example The Witness puzzles with size 1×2



(a) A 4×4 puzzle with 5 types of constraints

(b) The solution to the puzzle in (a)

Figure 2: Example 4×4 complex puzzle and its solution

MUSE and Probability

We denote the MUSE of puzzle X as $\mu(X)$. In the calculation of $\mu(X)$, it is assumed that the probability distribution of player actions is uniform, and $|A(s)|$ represents the number of available actions at each state s. Therefore, in the policy π_μ, $\pi_\mu(a, s) = |A(s)|^{-1}$ and the local entropy at s is $\log_2 |A(s)|$ (Chen, White, and Sturtevant 2023). Moreover, let $\tau_m = \{(s_i, a_i)\}_{i=0}^{m}$ be the sequence of state-action pairs starting at the root of the tree s_0 and ending at terminal state $T(s_m, a_m) = s$, then $\pi(s) = \prod_{(s_i,a_i)\in\tau_m} \pi(a_i, s_i)$ and $\pi(g)$ represents the probability of reaching the goal state g through the solution path, τ_g, according to π. Similarly, $\pi_\mu(g)$ is the probability of reaching the goal state g by uniform random. Now, we can derive the following theorem.

Theorem 1 *$\mu(X)$ is proportional to the probability of finding the most likely solution according to the player's policy (uniform random), where each solution path to the goal state $g \in G$ is a state-action sequence τ_g. It can be calculated by:*

$$\mu(X) = -\log_2 \max_{g\in G} \pi_\mu(g) \tag{3}$$

Proof. According to the recursive definition, μ of a non-terminal state, $\mu(s)$, is obtained by summing the local entropy and the minimum entropy of its successors. The local entropy is $\log_2 |A(s_i)|$ since $\pi_\mu(s) = |A(s_i)|^{-1}$. Hence, we have the following cases:

$$\mu(s) = \begin{cases} 0 & \text{if } s \in G \\ \min_{s_i\in\sigma(s)} \mu(s_i) + \log_2 |A(s_i)| & \text{if } A(s_i) \neq \varnothing \\ \infty & \text{otherwise} \end{cases}$$

By simplifying the definition without recursion, we obtain:

$$\mu(X) = \min_{g\in G} \sum_{s_i\in\tau_g} \log_2 |A(s_i)|$$

Icon and Name	Requirements
Must-Cross	The path must cross the constraint
Cannot-Cross	The path cannot cross the constraint
Separation	The constraint must be separated from different coloured separation constraints in the same region
Star	The containing area must have exactly one other region constraint with the same colour
Triangle	The path around it must occupy the same number of edges as the number of triangles

Table 1: Constraints and their requirements

where each solution path to the goal state $g \in G$ is a state-action sequence $\tau_g = \{(s_i, a_i) \mid T(s_i, a_i) = s_{i+1}$ and $T(s_{m_g}, a_{m_g}) = g\}_{i=0}^{m_g}$ and $s_i \in \tau_g$ stands for $s_i \in \{s_i \mid (s_i, a_i) \in \tau_g\}$.

Using the properties of logarithms, where $\log_2 x = -\log_2 x^{-1}$ and the logarithm function is monotone, we have:

$$\begin{aligned}
\mu(X) &= \min_{g\in G} \sum_{s_i\in\tau_g} \log_2 |A(s_i)| \\
&= \min_{g\in G} \log_2 \prod_{s_i\in\tau_g} |A(s_i)| \\
&= \min_{g\in G} \left[-\log_2 \prod_{s_i\in\tau_g} |A(s_i)|^{-1} \right] \\
&= \min_{g\in G} \left[-\log_2 \prod_{(s_i,a_i)\in\tau_g} \pi_\mu(a_i, s_i) \right] \\
&= \min_{g\in G} \left[-\log_2 \pi_\mu(g) \right] \\
&= -\log_2 \max_{g\in G} \pi_\mu(g)
\end{aligned}$$

Thus, MUSE is equivalent to the logarithm of the probability of the most likely solution path, given uniform probability on all actions. ■

Minimum Solution Information

Now we extend MUSE to cover the case where the player's policy is not uniform, and thus the local entropy is not $\log_2 |A(s)|$. For example, suppose $|A(s)| = 2$. This is like a fair coin flip, where getting heads represents taking the action leading to the goal. If the coin is fair, then the entropy of this event is $H = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$ Sh, which equals to the information content of each outcome. However, if the coin is not fair and the probability of heads

is 0.1, then the entropy of this event is $H = -0.1 \log_2 0.1 - 0.9 \log_2 0.9 \approx 0.469$ Sh. This results shows that tossing this unfair coin is less uncertain than tossing a fair one, since the outcome is highly unlikely to be heads. Therefore, the outcome of heads is more informative in this event, which is $I = -\log_2 P(\text{Head}) = -\log_2 0.1 \approx 3.322$ Sh. Similarly, if an oracle needs to describe the solution to the player in this case, it also needs to communicate 3.22 Sh to the player.

We generalize MUSE to handle arbitrary policies by defining the minimum solution information (MSI) of a puzzle X as $I^*(X)$ shown in Equation 4, which represents the minimum information needed to describe a specific goal from puzzle X.

$$I^*(X) = -\log_2 \max_{g \in G} \pi(g) \tag{4}$$

If player's policy is uniform, then $\pi = \pi_\mu$ and thus $I^*(X) = \mu(X)$.

Total Solution Information

Similar to ReMUSE, if a puzzle has multiple solutions (goal states) and the player does not have a preference for a specific one, then the oracle does not have to communicate an exact solution, but can minimize the expected communication needed to get the player to a goal. Considering the state-space tree of possible moves, minimizing the expected amount of information for every state in the tree requires minimizing the sum of the communication needed at that state and the expected amount of information necessary for each direct successor. For example, in MUSE or MSI, following the communication, the player's policy is updated such that $\pi'(a, s) = 1$ if and only if $s_a = T(a, s)$ is on the most likely solution path and 0 otherwise, as shown in Figure 3. Therefore, the amount of information required for the communication at the current node can be also calculated by $D_{KL}(\pi' \parallel \pi)$.

Recall that $\sigma(s)$ is the successor function for s. Then, we define the total solution information (TSI) at state s, $\psi(s)$, as

$$\psi(s) = \begin{cases} 0 & \text{if } s \in G \\ \min F(\pi'(s)) & \text{if } A(s) \neq \varnothing \\ \infty & \text{otherwise} \end{cases} \tag{5}$$

where F is defined as

$$F(\pi') = D_{\text{KL}}(\pi' \parallel \pi) + \mathbb{E}_{\pi'}\left[I\left(\sigma(s)\right)\right] \tag{6}$$

$D_{\text{KL}}(\pi' \parallel \pi)$ captures the amount of information utilized for shifting the player's policy from π to π' using KL divergence. $\mathbb{E}_{\pi'}\left[I\left(\sigma(s)\right)\right]$ is the expected amount of information required for successors of s under the new policy π'.

After carefully examining $\psi(s)$, we propose the following theorem:

Theorem 2 *$F(\pi'(s))$ is minimal if*

$$\pi'(a, s) = \frac{\pi(a, s) \cdot P_\pi(g_a \mid s_a)}{\sum_{a \in A(s)} \left[\pi(a, s) \cdot P_\pi(g_a \mid s_a)\right]} \tag{7}$$

and

$$\min F(\pi'(s)) = -\log_2 \sum_{a \in A(s)} \pi(a, s) \cdot P_\pi(g_a \mid s_a) \tag{8}$$

where $P_\pi(g_a \mid s_a)$ represents the probability of reaching a goal g_a from the successor state $s_a = T(s, a)$ of state s according to the player's original policy π. The proof for Theorem 2 is provided in (Shen 2024).

Equation 7 shows that the minimum amount of communication to change the policy to the one that always reaches a goal is multiplying the original policy with the probability of reaching a goal g_a from the successor s_a after taking the action a under the original policy π. Figure 4 presents an example of the behavior of the communication for TSI within a state-space tree, similar to the one depicted in Figure 3. Specifically, the original policy π is $[\frac{1}{3}, \frac{2}{3}]$ at s_0 and the probabilities for reaching g_0 and g_1 from s_1 and s_2 ($P_\pi(g_0 \mid s_1)$ and $P_\pi(g_1 \mid s_2)$) are $\frac{3}{4}$ and $\frac{1}{2}$, respectively. Therefore, after receiving the communication, the policy updates to $[\frac{1}{3} \times \frac{3}{4}, \frac{2}{3} \times \frac{1}{2}]$. Upon normalization, it becomes $[\frac{3}{7}, \frac{4}{7}]$.

Subsequently, the minimum amount of information required for communicating the solution is proportional to the probability of reaching a goal from the current state s, as shown in Equation 8. By considering the current state as the initial state s_0, we can thus simplify the definition without recursion as follows:

$$\psi(X) = -\log_2 \sum_{g \in G} \pi(g) \tag{9}$$

Therefore, the TSI of a puzzle, $\psi(X)$, is derived from the logarithm of the sum of the probabilities of reaching goals according to the player's policy π.

In contrast, ReMUSE employs a softmin function with base e for calculating the probability distribution of the uniform entropy of the immediate children (Chen, White, and Sturtevant 2023), which effectively shifts the distribution of reaching goals since uniform entropy is derived from the log of base 2 instead of e. Moreover, it is not possible to derive an equation for ReMUSE without recursion. Even if we know a player's policy and the corresponding probability of reaching the goal(s), calculating ReMUSE still requires recursively traversing the state space of the puzzle.

To summarize, MUSE computes the information needed from an oracle to find a *single* solution to a puzzle, given a uniform *a priori* policy. MSI generalizes this to find the information needed for a single solution given any *a priori* policy. TSI computes the minimum expected information needed to find *any* solution, given any *a priori* policy. TSI can be expressed more simply than ReMUSE, as TSI is just the logarithm of the probability of reaching the solution with a given policy.

Inference Rules

When solving logic puzzle games, players often deduce sets of inference rules, which are guidelines that can solve or simplify the puzzle (Chen, White, and Sturtevant 2023). Formally, an inference rule is a function that determines whether a specific action should be taken at a given state. When a player's policy incorporates inference rules, the probability of actions that cannot be taken is reduced to zero. Conversely, if an action must be taken, then the probability of that action is one, and the probability of all other

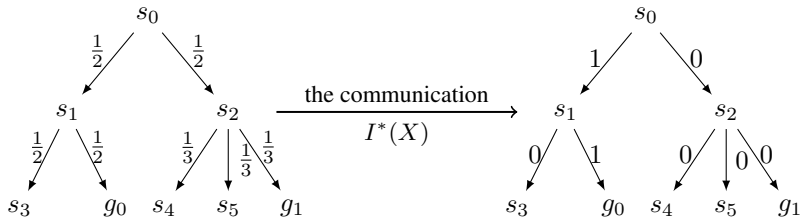


Figure 3: How the communication shifts the player's policy in MUSE and MSI

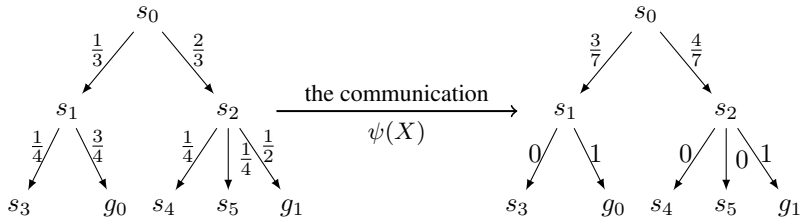


Figure 4: How the communication shifts the player's policy in TSI

actions is zero. For actions in Witness puzzles (a_{up}, a_{right}, a_{down}, a_{left}), each action can be classified as unknown (UNKNOWN), cannot be taken (CANNOT_TAKE), or must be taken (MUST_TAKE). These classifications reflect the probability of solving the puzzle by taking the corresponding actions. For instance, CANNOT_TAKE actions either directly violate the puzzle constraints or lead to situations where constraints cannot be satisfied later. Similarly, necessary actions to satisfy constraints are classified as MUST_TAKE. If there are multiple MUST_TAKE actions, then the puzzle is not solvable at that state since the player can take only one action at each state. Note that inferences rules can be directly incorporated into policies, but a policy cannot always be turned into an inference rule. It is useful to reason explicitly about inference rules separate from policies, as this can be used to model players. An example of integrating inference rules into π_μ is shown in Algorithm 1.

Chen, White, and Sturtevant (2023) proposed two inference rules for must-cross and separation constraints in their work, and another one for triangle constraints in a follow-up thesis (Chen 2023). We briefly explain these rules in Figure 5. Furthermore, we assume the state-space tree of a given puzzle mirrors that of an empty puzzle of the same size but without constraints. In such a tree, each path from the root to a leaf node represents a solution to the empty puzzle. This assumption generalizes the scenarios described by Chen (2023), where actions are classified as CANNOT_TAKE.

To expand the dataset for testing algorithms and study the effect of different player models, we propose the following additional inference rules that could be applied broadly across Witness puzzles containing different types of constraints.

Along-the-path Rule (APR)

For any sub-path (or subsequence) of a solution path that begins at the initial state, known as a partial solution, the region constraints that are adjacent to the same side of the path must not be violated. These constraints are guaranteed to remain in the same region regardless of the rest of the solution, allowing for immediate evaluation to determine the validity of the partial solution. For example, in Figure 6a, there are 8 star constraints adjacent the path, and none of them are violated. In contrast, there are 3 violated star constraints adjacent to the right side of the path in Figure 6b, thus this path is not part of any solution. The implementation of this inference rule is provided in Algorithm 2.

Region-completion Rule (RCR)

When the path intersects the boundary of the puzzle twice, it forms a new closed region. A closed region that does not contain the end junction cannot be changed, as shown in Figure 7. Every constraint within a closed region must be satisfied; otherwise, these constraints cannot be satisfied later. Algorithm 3 provides the details of this inference rule.

Experiments and Results

Our experiments aim to answer the following questions: First, how does TSI perform compared to ReMUSE on different puzzle datasets? Second, how do policies from machine learning (ML) policy compare to our manually created inference rules? Third, how does the quality of the player model affect the puzzle difficulty calculation and its correlation with player enjoyment?

Experiment Setup

The Windmill (Gruen 2016) is a website where users can design and publish Witness puzzles. It also allows users to upvote or downvote puzzles based on their enjoyment. We assume that users on the website are advanced players of *The Witness* and they generally upvote challenging puzzles and downvote trivial ones (Chen, White, and Sturtevant 2023).

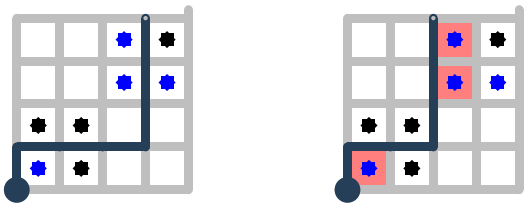
(a) a_{right} is MUST_TAKE, since a must-cross constraint is adjacent to the current junction

(b) a_{right} is MUST_TAKE, since two separation constraints are adjacent to each other and the path must go between them

(c) a_{right} is MUST_TAKE, since the double triangle cannot be satisfied later by taking a_{up}

(d) a_{right} is CANNOT_TAKE, since the triple triangle cannot be satisfied later by taking it

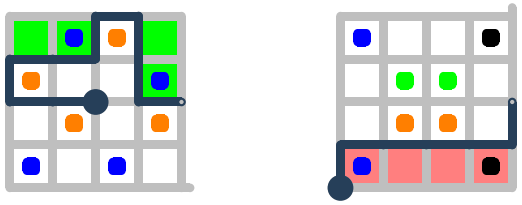
Figure 5: The baseline inference rules proposed in previous studies (Chen, White, and Sturtevant 2023; Chen 2023)



(a) A puzzle with a path that does not violate any region constraints next to it

(b) The same puzzle as (a) but with a path that violates star constraints highlighted in red on its right-hand side

Figure 6: Example puzzles with similar paths



(a) A puzzle with 2 completed regions highlighted in green, where all separation constraints inside them are satisfied

(b) A puzzle with 1 completed region highlighted in red, where all separation constraints inside it are not satisfied

Figure 7: Example puzzles with completed regions highlighted

The complete puzzle dataset comprises every 4×4 puzzle prior to November 29, 2022, consistent with previous studies (Chen, White, and Sturtevant 2023). This fixed size limits the maximum difficulty and eliminates the variable of puzzle size. We excluded all puzzles that are included in the original game or duplication, applying different filters to obtain subsets of puzzles for various experiments. Additionally, the number of upvotes for each puzzle is normalized based on the linear relationship between upvotes and timestamps within the test set to account for temporal variations, as the number of active users on the website decreases over time. Furthermore, for reproducibility, we implemented MUSE, ReMUSE and inference rules from the previous study (Chen, White, and Sturtevant 2023; Chen 2023) individually and assumed player policies to be uniform after applying inference rules.

To assess the relationship between player enjoyment and puzzle difficulty, as well as the quality of difficulty assessment, we compare the Pearson Correlation Coefficients (r-values) derived from simple linear regression of puzzle difficulty and the number of upvotes. The corresponding probabilities of the F-statistic (p-values) represent the likelihood of failing to reject the null hypothesis; in other words, they indicate that the variable has no effect.

Comparing Different Algorithms

The first test set is identical to the one used in the previous study (Chen, White, and Sturtevant 2023). It contains 104 puzzles that incorporate only separation and path constraints. Table 2 shows the correlation computed from different algorithms, and Figure 8 provides details of the difficulty score of each puzzle and its linear correlation with the number of upvotes. In this case MSI is identical to MUSE, because a uniform default policy is assumed. We observed that TSI and ReMUSE, however, are not the same, but TSI is very similar to ReMUSE on this test set.

Approach	Correlation	p-value
MUSE (Reported)	0.410	1.10×10^{-5}
MUSE/MSI (Reproduction)	0.412	1.53×10^{-5}
ReMUSE (Reported)	0.570	1.57×10^{-10}
ReMUSE (Reproduction)	0.583	1.02×10^{-10}
TSI	0.563	5.78×10^{-10}

Table 2: Correlation and p-value for different approaches evaluated on the dataset used in previous study (Chen, White, and Sturtevant 2023)

Applying Additional Inference Rules

The second test set (T-only set) comprises 179 puzzles, each containing at least one triangle constraint and possibly one or both path constraints. The first half of Table 3 provides details of how inference rules affect the correlation. The absolute correlations without applying any additional inference

Algorithm 1 π_μ with inference rules

```
function π_μ(a, s)
    Initialize empty map M
    for each a' ∈ A(s) do
        M[a'] = UNKNOWN
    end for
    for each f ∈ {All inference rules} do
        if f(a, s) = CANNOT_TAKE then
            return 0
        else if f(a, s) = MUST_TAKE then
            return 1
        end if
        for each a' ∈ A(s) do
            if M[a'] ≠ UNKNOWN and
            M[a'] ≠ f(a', s) then
                return 0          ▷ conflicts between rules
            end if
            M[a'] ← f(a', s)
        end for
    end for
    A' ← ∅
    for each a' ∈ A(s) do
        if M[a'] = MUST_TAKE and a' ≠ a then
            return 0
        else if M[a'] = CANNOT_TAKE then
            continue
        end if
        A' ← A' ∪ {a'}            ▷ append a' to A'
    end for
    return |A'|^-1
end function
```

Algorithm 2 Along-the-path Rule

```
function PATHTEST(s)
    ▷ Record blocks on the left-hand side and right-hand
      side of the path that are not in a completed region,
      into lists lhs and rhs, respectively. ◁
    lhs, rhs ← GETLEFTRIGHTBLOCKS(s)
    if ∃c_region ∈ lhs and c_region is not satisfied then
        return false
    end if
    if ∃c_region ∈ rhs and c_region is not satisfied then
        return false
    end if
    return true
end function

function ALONGTHEPATHRULE(a, s)
    Apply Action a to s
    r ← PATHTEST(s)
    Undo Action a
    if r = true then
        ▷ cannot be determined by this rule ◁
        return UNKNOWN
    else
        return CANNOT_TAKE
    end if
end function
```

rule are lower than in previous test sets, as expected (Chen, White, and Sturtevant 2023). More importantly, adding the along-the-path rule improves the correlation, while adding the region-completion rule reduced correlation. This may indicate that players tend to use along-the-path rules when the puzzle contains triangles.

For comparison, we conduct a follow-up experiment on the complementary set of the second test set (non-T set): the set of 226 puzzles that do not contain triangle constraints (containing separation constraints, star constraints, and path constraints). The second half of Table 3 demonstrates that incorporating more inference rules, resulting in a more informed model, does not necessarily lead to a better correlation between difficulty scores and player upvotes. One might infer that users on *The Windmill* are not necessarily using all of the inference rules that we developed. However, as we discuss further at the end of the experimental results, there may be other factors in play: good puzzles will have inference rules that help guide players to the solution. Regardless, it is crucial to model player knowledge accurately when measuring puzzle difficulty.

Difficulty Calculated by ML Policy

Since MSI and TSI are derived from the probability of reaching goals, it is possible to use a policy provided by a machine learning (ML) model instead of manually creating inference rules. Lelis et al. (2022) utilized a modified Bootstrap model (Jabbari Arfaee, Zilles, and Holte 2011), which takes a puzzle instance as input and returns a policy, to be used as a policy in Levin tree search (Orseau et al. 2018) for generating an equidistant curriculum of specific types of Witness puzzles. This curriculum is designed to ease player learning. The model was trained on Witness puzzles that contain only separation constraints with two colors, referred to as Black and White Square (BWS) puzzles. The authors shared the source code and corresponding datasets with us, and we trained their model independently. The complete dataset contains 24 BWS puzzles. We collect all solutions for each of them. To compute the MSI, we calculate the logarithm of the probability of reaching the most likely solution as dictated by the model's policy using Equation 4. For the TSI, we calculate the logarithm of the sum of the probabilities of reaching each solution (goal state) using Equation 9. The baseline inference rule in these puzzles is the separation rule (SR), explained in Figure 5b, since other inference rules proposed previously (Chen 2023) do not affect puzzles in this test set. The results shown in Table 4 indicate that the MSI and TSI derived from the ML policy are very close to each other. Notably, the TSI derived from the ML policy surpasses that derived from inference rules on these simple puzzles. Additionally, the results show that applying extra inference rules worsens the correlation, consistent with the previous experiment.

To verify that the model learns inferences, we conduct a supplementary experiment. We first collect states along all solution sequences of each puzzle that have actions pruned

Algorithm 3 Region-completion Rule

```
function RegionTest(s)
    R ← GetCompletedRegions(s)
    for each r ∈ R do        ▷ Check each completed region
        if ∃c_region ∈ r and c_region is not satisfied then
            return false
        end if
    end for
    return true
end function

function RegionCompletionRule(a, s)
    Apply action a to s
    if a region is completed then
        r ← RegionTest(s)
    else
        r ← true
    end if
    Undo action a
    if r = true then        ▷ Cannot be determined by this rule
        return UNKNOWN
    else
        return CANNOT_TAKE
    end if
end function
```

Puzzle Set	Inference Rule	Correlation	p-value
T-only	Baseline	0.366	4.67×10^{-7}
T-only	APR	**0.433**	1.40×10^{-9}
T-only	RCR	0.252	6.58×10^{-4}
T-only	Both	0.322	1.09×10^{-5}
non-T	Baseline	**0.515**	1.03×10^{-16}
non-T	APR	0.494	2.62×10^{-15}
non-T	RCR	0.497	1.71×10^{-15}
non-T	Both	0.479	2.16×10^{-14}

Table 3: Correlation and p-value for for different player models evaluated on triangle-only and non-triangle Witness puzzles

by the separation rule shown in Figure 5b. We then calculate the expected probability of the policy taking CANNOT_TAKE actions. Additionally, we calculate the expected probability of taking the correct action, as the separation rule selects actions that are MUST_TAKE rather than explicitly identifying other actions as CANNOT_TAKE. In general, if the policy learns the inference that prunes actions, then the probability of taking these pruned actions should be close to 0. Similarly, if the policy learns the inference that selects actions, then the probability of taking these selected actions should be close to 1. Furthermore, we collect states along all solution sequences of every puzzle that the separation rule cannot be applied but other two new inference rules can be applied respectively. The results are provided in Table 5. They show that the model adequately learned the along-the-path rule; however, it did not learn the region-completion

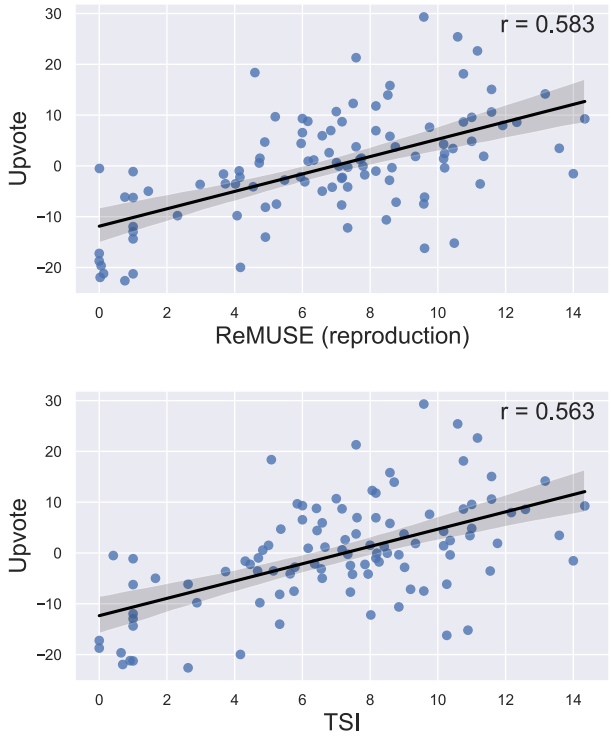


Figure 8: The number of upvote of puzzles from the dataset used by the previous study (Chen, White, and Sturtevant 2023) and their difficulty measured by ReMUSE and TSI respectively

rule effectively, which is also reflected in Table 4. Consequently, we conclude that the model is capable of learning inference and can be effectively utilized in both MSI and TSI.

Integration with EPCG

Procedural Content Generation (PCG) is a technique that involves the algorithmic creation of content rather than manual creation, aiming to reduce content production costs and enable the production of vast, diverse, and dynamic environments, characters, and scenarios (De Kegel and Haahr 2020). Exhaustive PCG (EPCG) extends this concept by striving to explore all possible permutations and variations within a defined set of parameters (Sturtevant and Ota 2018).

As part of this overall work, we developed a puzzle editor that allows designers to place question marks on the puzzle, as shown in Figure 9c. The generator will exhaustively try all possible constraints at these locations and return the resulting solvable puzzles sorted by TSI. In March 2024, we selected the latest puzzle (Figure 9a) posted on *The Windmill* and used our editor to generate a similar puzzle with a higher TSI (Figure 9b). We then published it alongside the original one. Three months later, the original puzzle had 207 solves and 4 upvotes, while our puzzle had only 113 solves and 5 upvotes.

In going back and trying to solve this puzzle again, we

Player Model	MSI		TSI	
	Correlation	p-value	Correlation	p-value
SR (baseline)	0.521	8.99×10^{-3}	**0.844**	2.23×10^{-7}
APR	0.401	5.23×10^{-2}	0.720	7.20×10^{-5}
RCR	0.405	4.94×10^{-2}	0.449	2.77×10^{-2}
SR and APR	0.341	1.03×10^{-1}	0.769	1.15×10^{-5}
SR and RCR	0.490	1.51×10^{-2}	0.827	6.26×10^{-7}
APR and RCR	0.377	6.92×10^{-2}	0.739	3.69×10^{-5}
All Inference Rules	0.299	1.55×10^{-1}	0.771	1.02×10^{-5}
ML Policy	**0.751**	2.35×10^{-5}	0.787	5.16×10^{-6}

Table 4: Correlation and p-value for MSI and TSI with different player models evaluated on BWS Witness puzzles (Lelis et al. 2022)

Method	Probability
uniform	0.497
policy	0.002
inference	0.000

(a) Expected probabilities of taking CANNOT_TAKE actions pruned by the separation rule in total of 11050 states

Method	Probability
uniform	0.497
policy	0.994
inference	1.000

(b) Expected probabilities of taking MUST_TAKE actions selected by the separation rule in total of 11050 states

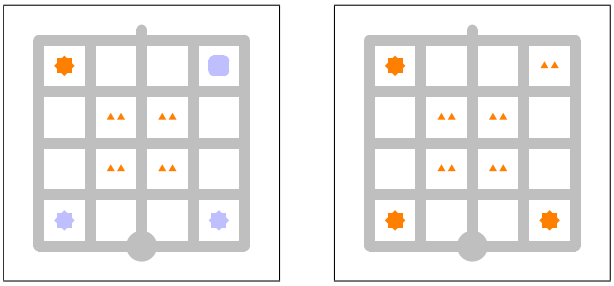
Method	Probability
uniform	0.465
policy	0.091
inference	0.000

(c) Expected probabilities of taking CANNOT_TAKE actions pruned by the along-the-path rule in total of 4518 states

Method	Probability
uniform	0.418
policy	0.331
inference	0.000

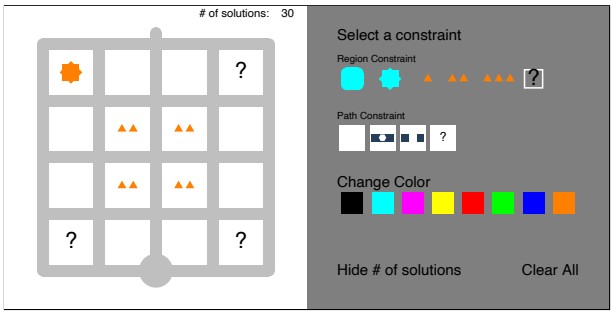
(d) Expected probabilities of taking CANNOT_TAKE actions pruned by the region-completion rule in total of 2998 states

Table 5: Methods and the expected probabilities of taking different actions



(a) Original puzzle with TSI of 6.67 Sh

(b) Generated puzzle using our editor with TSI of 8.58 Sh



(c) GUI of our editor

Figure 9: Screenshots of puzzles and the editor

find that it is difficult for us to solve, because there are very few inferences we can make to guide our own problem-solving process. We end up solving it more by random exploration than by reasoned exploration. TSI is high because no inference rules reduce the search. Our experience is reflected in the relative low total number of solves.

We must be careful not to draw too broad of conclusions from this single puzzle. This result suggests that instead of just looking for puzzles with high TSI, we may be interested in finding puzzles that have a large difference in TSI depending on the inference rules used. This would indicate puzzles that can be more easily solved if you know the inference rule, but are hard otherwise. This approach has been used recently in curriculum generation (Mahmoud and Sturtevant 2024). Regardless, more work is needed to understand the interplay between TSI, player models, and puzzle enjoyment.

Conclusion

In this paper, we generalized entropy and solution information and proposed new algorithms, MSI and TSI, as general metrics for measuring puzzle difficulty. These novel approaches facilitate analysis involving both inference rules and player policies. After validating these metrics with two additional inference rules through extensive experiments on various puzzle sets and player models, we conclude that our approach is as effective as previous methods while offering an improved mathematical foundation and generality. Furthermore, we demonstrated that accurate player modeling is essential. Finally, we developed a Witness puzzle editor that supports EPCG queries.

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 Csíkszentmihályi, M. 2012. The Importance of Challenge for the Enjoyment of Intrinsically Motivated, Goal-Directed Activities. *Personality & social psychology bulletin*.

CandyCane Software. 2011. Fling! https://www.candycaneapps.com/fling/. Accessed: 2024-06-15.

Chen, E. Y. C. 2023. *Entropy as a Measure of Puzzle Difficulty*. Master's thesis, University of Alberta.

Chen, E. Y. C.; White, A.; and Sturtevant, N. R. 2023. Entropy as a Measure of Puzzle Difficulty. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 19(1): 34–42.

De Kegel, B.; and Haahr, M. 2020. Procedural Puzzle Generation: A Survey. *IEEE Transactions on Games*, 12(1): 21–40.

del Solar-Zavala, J. A.; Schaa, H.; and Barriga, N. A. 2023. Using Entropy for Modeling Difficulty in the Asteroid Escape Sliding Puzzle. In *2023 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*. IEEE.

Gruen, M. 2016. The Windmill. https://windmill.thefifthmatt.com. Accessed: 2024-06-15.

International Organization for Standardization. 2008. Quantities and units – Part 13: Information science and technology. https://www.iso.org/standard/31898.html. Accessed: 2024-06-15.

Jabbari Arfaee, S.; Zilles, S.; and Holte, R. C. 2011. Learning heuristic functions for large state spaces. *Artificial Intelligence*, 175(16–17): 2075–2098.

Jarušek, P.; and Pelánek, R. 2010. Difficulty Rating of Sokoban Puzzle. In *Proceedings of the 2010 Conference on STAIRS 2010: Proceedings of the Fifth Starting AI Researchers' Symposium*, 140–150. NLD: IOS Press. ISBN 9781607506751.

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

Lelis, L. H. S.; Nova, J. G. G. V.; Chen, E. Y. C.; Sturtevant, N. R.; Epp, C. D.; and Bowling, M. 2022. Learning Curricula for Humans: An Empirical Study with Puzzles from The Witness. In *International Joint Conference on Artificial Intelligence*.

Mahmoud, Y.; and Sturtevant, N. R. 2024. Using EPCG for Designing a Hexagon Tangram Puzzle. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Nelson, M. J.; and Smith, A. M. 2016. *ASP with Applications to Mazes and Levels*, 143–157. Springer International Publishing. ISBN 9783319427164.

Noumenon Games. 2015. Snakebird. https://store.steampowered.com/app/357300/Snakebird/. Accessed: 2024-06-15.

Orseau, L.; Lelis, L. H. S.; Lattimore, T.; and Weber, T. 2018. Single-agent policy tree search with guarantees. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 3205–3215. Red Hook, NY, USA: Curran Associates Inc.

Roohi, S.; Guckelsberger, C.; Relas, A.; Heiskanen, H.; Takatalo, J.; and Hämäläinen, P. 2021. Predicting Game Difficulty and Engagement Using AI Players. *Proc. ACM Hum.-Comput. Interact.*, 5(CHI PLAY).

Rovio Entertainment Corporation. 2009. Angry Birds. https://www.angrybirds.com/. Accessed: 2024-06-15.

Shannon, C. E. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3): 379–423.

Shen, J. 2024. Generalized Entropy and Solution Information for Measuring Puzzle Difficulty. Forthcoming.

Smith, A. M.; Andersen, E.; Mateas, M.; and Popović, Z. 2012. A case study of expressively constrainable level design automation tools for a puzzle game. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG'12. ACM.

Sturtevant, N. 2021. An Argument for Large-Scale Breadth-First Search for Game Design and Content Generation via a Case Study of Fling! *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 9(3): 28–33.

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. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 14(1): 109–115.

Thekla, Inc. 2016. The Witness. https://store.steampowered.com/app/210970/The_Witness/. Accessed: 2024-06-15.

van Kreveld, M.; Loffler, M.; and Mutser, P. 2015. Automated puzzle difficulty estimation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE.

ZeptoLab UK Limited. 2010. Cut The Rope. https://www.cuttherope.net/. Accessed: 2024-06-15.