Dynamic Crossword Difficulty via Reactive Puzzle Construction

Colan F. Biemer
Khoury College of Computer Sciences
Northeastern University
Boston, USA
bi3mer93@gmail.com

Seth Cooper
Khoury College of Computer Sciences
Northeastern University
Boston, USA
se.cooper@northeastern.edu

Abstract

Traditionally, crosswords are made as complete puzzles and then shared with players. In the digital age, crosswords can be created dynamically while the player plays, and can even react to how the player is performing. In this paper, we present a system for reactive crossword construction as a form of dynamic difficulty adjustment. Difficulty can be adapted within a single puzzle, by constructing a puzzle while the player plays, and the word/clue pairs can be chosen to be more or less difficult. We also show that hints, in the form of additional words, can be embedded into the puzzle for struggling players. We tested this system with three player personas across five levels of puzzle difficulty, using time as a proxy for difficulty. Our results show that reactive puzzle construction reduced the time needed to solve puzzles compared to static construction. The addition of hints further reduced the time to solve. These results are limited to simulated player personas, but establish a foundation and identify directions for future evaluation with real players. We expect that reactive puzzle construction could be applied to otherwise static puzzle types, by revealing clues as players progress.

Keywords

crossword, puzzle, dynamic difficulty adjustment, reactive puzzle construction

ACM Reference Format:
Colan F. Biemer and Seth Cooper. 2026. Dynamic Crossword Difficulty via Reactive Puzzle Construction. In *Foundations of Digital Games (FDG '26), August 10–13, 2026, Copenhagen, Denmark*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3815598.3815714

1 Introduction

The first crossword puzzle was made in 1913 [21], and has become an enduring part of our culture.[1] The structure of a crossword puzzle is a grid that can be filled in with letters to form words based on a set of clues for each row and column. The puzzle is, thus, static, meaning it never changes. Puzzles are created, hopefully tested, and then sent off into the world for others to play.

The static nature of a crossword is a feature, not a bug. Many revel in the difficulty of harder puzzles like cryptic crosswords, but there is more to crosswords than challenge. Crosswords can be used to promote learning [5, 14]. If clues are too easy, the player learns nothing. If clues are too confusing (i.e. the solver cannot figure out the answer), then the player may enter a state of unresolved confusion, which is linked with worse learning outcomes than partial confusion [4]. As a result, a static crossword designed for learning can be improved by adding dynamic elements to help stuck players.

A digital crossword can be adapted while the player is playing, reacting to their performance, and changing upcoming words and clues. To explore this, we created a crossword game with three different modes: Static, Reactive, and Reactive with Hints. The Static mode generates puzzles whole cloth. The Reactive mode adds new words to the puzzle as the player solves it and reduces the expected difficulty if the player appears to be stuck. Reactive with Hints is the same, except it offers players a hint if they get stuck, where a hint adds a new, easy word to the crossword grid that connects with the challenging word. We evaluated these three game modes with player personas across five puzzle difficulty levels.

Personas struggled most with statically generated puzzles, spending the most time on them on average. Hint-free reactive generation was harder than hint-assisted, but personas encountered fewer challenging words as a result.

Our contributions are the following: (1) A real-time adaptive puzzle generation system applied to crosswords that uses word surprisal (see Section 3.1) to dynamically adjust puzzle difficulty [8, 11]; (2) a hint method for placing easy words with intersecting hard words to reduce challenge; and (3) a playable, open source implementation.[3]

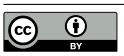
2 Related Work

Crossword generation is not a novel contribution of this work. As early as 1990, work in the field framed it as a constraint satisfaction problem [6]. Further work has gone into generating puzzles that match a specific structure [1]. This work does not advance the area of crossword generation to fit a given structure, which is a non-trivial task due to the size of the search space and the sparseness of solutions [2]. Instead, the game built for this work uses an algorithm that finds a valid placement but makes no attempt to fit a given structure.

A substantial amount of effort has also gone into generating datasets of word/clue pairs. For example, Rigutini et al. developed a method that scraped the web, then used NLP to break down the results, and then built a word/clue dataset [15]. Zugarini et al. simplified the NLP step by using large language models [22], and this work uses the dataset they created.

[1] Fun fact: the original name was not "Crossword," it was "Word-Cross." The name changed due to a typesetting error, and the new (better[2]) name stuck [12].
[2] Subjectively.

[3] https://github.com/bi3mer/crossword



FDG '26, Copenhagen, Denmark

ACM ISBN 979-8-4007-2495-4/2026/08
https://doi.org/10.1145/3815598.3815714

Work has also gone into developing solvers for crosswords. For example, the Berkeley solver [19] generates candidate answers for word/clue pairs based on words that fit (i.e., the right number of characters) with a neural question answering model and then runs a local search to find full puzzle solutions. Work in this area, though, is not related to how we use player personas. Our personas do not solve the puzzle so much as estimate how long it would take a player of a certain skill level to solve it.

Dynamic difficulty adjustment (DDA) alters the game based on the player [10]. If, for example, the player is having no trouble at all, a game with DDA will get harder. Alternatively, if the player is in a constant state of intense, frustrating challenge, a game with DDA will get easier. How DDA is accomplished, though, depends on the game. Hunicke described one approach that modulated in-game systems in a combat game based on the player's probability of dying.

Whereas Hunicke applied DDA to a first-person shooter, DDA can also be applied to other types of games. González-Duque et al. [7] applied fast content adaptation via Bayesian optimization to Sudoku—also a grid-based puzzle game like a crossword—where the goal was to make a level which took time t to solve. Their approach generated complete puzzles for the player to solve and is analogous to a more intelligent implementation of the Static game mode used in this work. Other work has also shown how sequences of puzzles can be adapted to improve outcomes [17]. In comparison, the work in this paper tries to build the puzzle as the player plays, adapting *within* the puzzle.

Closer to our work is *Absurdle* [9], a reactive version of *Wordle* [20]. The target word changes based on player input to maximize the ambiguity of clues given from input. Where we want puzzles to be of an appropriate difficulty for the target player, *Absurdle* tries to make the target word as hard as possible.

3 Approach

3.1 Dataset

We used Clue-Instruct [23] as a dataset for word/clue pairs. To model difficulty, we used *surprisal*, which is the negative log probability of an event. We modeled word probability using unigrams, calculated with `ngrams.dev` [18], which uses Google's N-Grams Dataset [13]. This choice has limitations. A common word yields low surprisal, but what if the clue were challenging? This would cause an erroneous difficulty estimate. A more complete model would compute surprisal conditioned on the clue [11]. We adopt word-level surprisal here as an approximation, since we are not evaluating against real players.

The resulting dataset contains 43,475 word/clue/surprisal tuples with surprisal values ranging from 1.32 to 26.74 (mean 16.36). Duplicate words have identical surprisal, but different clues. The dataset was sorted by surprisal (lowest to highest) to order words by increasing difficulty.

3.2 Crossword Implementation Introduction

The core of reactive crossword puzzle construction is an algorithm that selects a word and places it on the grid to connect with existing words (which may or may not have been filled in). The full details of the implementation are out of scope for this work. However, a few details are necessary to understand the remaining sections.

To select words, we use a *difficulty index* (D_i), which is an index into the array of words sorted by surprisal, described in Section 3.1. For a new player, D_i starts at 0. However, in this work, we used different puzzle difficulties, as described in Section 3.5, which alter the starting value of D_i. To select a word, the word at the current difficulty index is chosen. However, for variety in puzzles generated, we added a 20% chance that the word at D_i is skipped, and in this case D_i is incremented to the next word in the dataset. Additionally, we found that three-letter words could cause problems when placed into the grid, so we added a check to skip them.

Once a word not already in the puzzle is selected, it is placed into the grid. The algorithm developed to do this attempts to find a place with the most valid intersections. Other than this, the algorithm is essentially a brute force solver that places words but makes no attempt to construct a symmetrical crossword.

If placement fails, then D_i is incremented and a new word is attempted to be placed. The search is exhaustive. Once placement was successful, D_i is incremented. This increases the difficulty, but the effect is negligible in a large dataset.

3.3 Puzzle Modes

Every puzzle generated has exactly 20 word/clue pairs, but when they are placed differs between modes.

Static — The static puzzle mode is a traditional crossword puzzle where the entire puzzle is present from the start. It uses the placement algorithm described in Section 3.2 twenty times in a row, incrementing D_i after each word is successfully placed.

Reactive — A reactive puzzle starts with two words already placed. When the player solves one, a new word is added. If a word has not been correctly filled in after 60 seconds, the system reduces the D_i by 10%, so that the next word added to the puzzle will be easier. This reduction is triggered at most once per period of inactivity; if no word was completed after twenty minutes, D_i would only be reduced once. It resets when the player solves a word, allowing it to trigger again on the next period of difficulty.

Reactive with Hints — This mode extends the Reactive mode with a hint system. If 30 seconds pass without the player solving a word, a hint is added to the puzzle. A hint is a non-duplicate, low-surprisal word (an index in the dataset starts at 0, and non-duplicate words are then tested for valid placements). The layout algorithm prioritizes placing this word on the entry the player has been struggling with, tracked via a per-entry timer. If the hint cannot be placed on the struggling entry, the system tries other unsolved entries. If no placement is possible, or the maximum number of word/clue pairs has been reached, no hint is added. If successful, the newly placed word fills in the cross-letters in the struggling entry, reducing its effective surprisal (s_{eff}) and making it easier to solve for the persona; see Equation 1: s is the original surprisal, l is the word length, and f is the number of filled cross-letters. A nonlinear approximation of s_{eff}, where filling in more letters yields a greater reduction in surprisal, may provide a more realistic model.

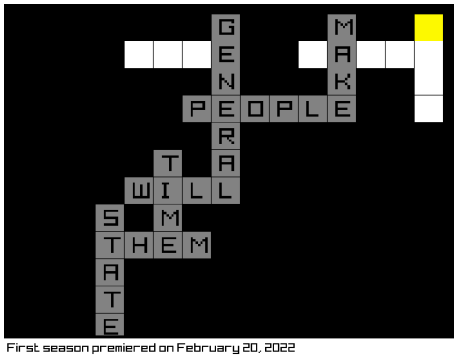


Figure 1: Example Reactive with Hints crossword puzzle in the process of being solved with a hint added to the word on the top right. The solution to the puzzle is "From."

$$s_{\text{eff}} = s\left(1 - \frac{f}{l}\right) \tag{1}$$

The player persona must solve a non-hint word before another hint can be added. Additionally, hints do add time to the puzzle because they have to be solved. See Figure 1 for an example puzzle with a hint added.

3.4 Player Personas

Table 1: Player persona configurations, using surprisal as a threshold for challenge outcomes with clues.

Persona	Solve Threshold	Struggle Threshold
Beginner	5.0	8.0
Intermediate	8.0	12.0
Expert	12.0	20.0

To evaluate the three puzzle modes, we made three player personas: *beginner*, *intermediate*, and *expert*. Each is defined by a pair of surprisal thresholds (Table 1). Given a word with surprisal s_{eff} in the crossword puzzle, the persona produces one of three outcomes: if s_{eff} is below the solve threshold, the word is *solved* quickly (1–10s); if s_{eff} falls between the solve and struggle thresholds, the persona *struggles* but eventually succeeds (30–60s); and if s_{eff} exceeds the struggle threshold, the word is too *hard* and the persona fails and has to, theoretically, go to the web to find the answer (120–300s). Within each outcome's time range, the simulated duration is linearly interpolated based on where s_{eff} falls relative to the threshold boundaries. Note that these time ranges are not empirically derived; they are approximations used to evaluate personas.

In terms of solving, personas take a linear approach, solving one word at a time. This can make some later words easier by filling in letters, shown by Equation 1. The mode that takes advantage of this is Reactive with Hints, where personas will stop simulating time on their current word/clue pair and instead solve the easier hint before going back to the original word/clue pair.

3.5 Puzzle Difficulty

Table 2: Puzzle difficulty levels and their starting surprisal.

Difficulty	Starting Surprisal
Very Easy	1.32
Easy	6.40
Medium	11.49
Hard	16.57
Very Hard	21.66

Every player persona played five different puzzle difficulties: very easy, easy, medium, hard, and very hard. Difficulty was based on surprisal, and the initial surprisal for each puzzle is shown in Table 2. This was used to initialize D_i, where the index into the dataset with the closest surprisal was used. These exact values were found by using the minimum and maximum surprisal values.

4 Results and Discussion

Table 3: Average surprisal range per puzzle by difficulty and game type across personas (see Table 1).

Difficulty	Game Type	Min	Avg	Max
Very Easy	Static	4.81	5.63	6.45
	Reactive	4.81	5.63	6.45
	Reactive+Hints	4.56	5.56	6.55
Easy	Static	6.42	6.60	6.77
	Reactive	6.42	6.60	6.77
	Reactive+Hints	5.63	6.19	6.75
Medium	Static	11.49	11.50	11.50
	Reactive	10.69	11.13	11.58
	Reactive+Hints	6.52	9.01	11.50
Hard	Static	16.57	16.57	16.58
	Reactive	13.48	15.03	16.57
	Reactive+Hints	4.03	10.30	16.57
Very Hard	Static	21.66	21.67	21.67
	Reactive	14.55	18.11	21.66
	Reactive+Hints	4.03	12.85	21.66

Each player persona solved each puzzle mode 1,000 times. Table 3 shows the minimum, maximum, and average surprisal for each generated puzzle. These values should be interpreted relative to the persona thresholds in Table 1; for example, a beginner struggles above 8.0 surprisal and an expert above 20.0. There is a discrepancy between Tables 2 and 3, where one would expect the minimum surprisal for Very Easy to be 1.32, but it is 4.81. This is because the placement algorithm requires words to be longer than three letters, and the word with a surprisal of 1.32 is "the."

In the very easy puzzle mode, Static and Reactive puzzle construction yielded identical surprisal values. However, Reactive with Hints had a lower minimum and average surprisal, but a higher maximum. We attribute the higher maximum surprisal to random

chance. Reactive had a lower minimum surprisal because the beginner player persona used hints. By reducing difficulty for struggling players, personas encountered fewer challenging words.

The remaining puzzle types show a similar pattern with Static having higher surprisal, Reactive with Hints having lower surprisal, and Reactive being in the middle. Reactive having lower surprisal than Static is due to the behavior of reducing the internal difficulty index by 10% when personas spend more than 60 simulated seconds solving a word.

Table 4: Average Puzzle Time (seconds) with Standard Deviation by Difficulty and Game Type

Persona	Puzzle Difficulty	Static	Reactive	Reactive + Hints
Beginner	Very Easy	421 ± 30	421 ± 30	**312 ± 28**
	Easy	577 ± 32	577 ± 32	**313 ± 24**
	Medium	2714 ± 32	2163 ± 70	**1176 ± 52**
	Hard	3530 ± 19	3043 ± 31	**1702 ± 34**
	Very Hard	4308 ± 24	3361 ± 34	**1943 ± 21**
Intermediate	Very Easy	**132 ± 1**	**132 ± 1**	**132 ± 1**
	Easy	**141 ± 1**	**141 ± 1**	**141 ± 1**
	Medium	894 ± 16	894 ± 16	**212 ± 15**
	Hard	2970 ± 41	1918 ± 43	**953 ± 49**
	Very Hard	3932 ± 30	2511 ± 79	**1373 ± 51**
Expert	Very Easy	**94 ± 1**	**94 ± 1**	**94 ± 1**
	Easy	**101 ± 1**	**101 ± 1**	**101 ± 1**
	Medium	**170 ± 1**	**170 ± 1**	**170 ± 1**
	Hard	814 ± 14	814 ± 14	**176 ± 8**
	Very Hard	1813 ± 93	1128 ± 10	**417 ± 44**

The time each persona took to solve the puzzles is shown in Table 4. While the personas were solving words with s_{eff} beneath their solve threshold, there was no difference between puzzle modes, as shown by the first two rows for intermediate and expert player personas in the table. However, once this was no longer the case, personas began using hints in the Reactive with Hints mode, and their overall time to solve the puzzle decreased.

Table 5: Average hints used per puzzle by difficulty and persona (Reactive + Hints mode only) with Standard Deviation.

Difficulty	Beginner	Intermediate	Expert
Very Easy	5.6 ± 0.8	0.0 ± 0.0	0.0 ± 0.0
Easy	7.0 ± 0.7	0.0 ± 0.0	0.0 ± 0.0
Medium	9.0 ± 0.0	9.0 ± 0.1	0.0 ± 0.0
Hard	9.0 ± 0.0	9.0 ± 0.0	9.0 ± 0.0
Very Hard	9.0 ± 0.0	9.0 ± 0.0	9.0 ± 0.0

Table 5 shows the number of hints used by difficulty. We can see that, unsurprisingly, the beginner needed hints on every puzzle. For the Very Easy and Easy puzzles, though, it did not reach the maximum of nine hints. Nine was the maximum because every puzzle starts with two word/clue pairs, and the system requires a word to be solved before another hint becomes available. As a result, hints and hard words strictly alternate, giving nine hard word/clue pairs and nine hints for a total of twenty word/clue pairs.

The Intermediate persona needed hints starting on the Medium difficulty puzzles. The Expert persona only needed hints for Hard and Very Hard puzzles, but we can also see that the puzzles became too challenging at the Hard difficulty since every persona had an average number of hints per puzzle of nine.

5 Conclusion

In this work, we showed how crossword puzzles can be constructed as they are solved and how the puzzle can react to the player's performance. The method was tested with three player personas, five puzzle difficulties, and three methods for generating crosswords. We found that player personas spent the most time on the static puzzles and the least time on a Reactive mode that embedded hints into the crossword itself.

While this work was limited to crosswords, we believe this kind of approach can be applied to other puzzles. Similar to fitting new words into an existing crossword grid, as long as the reactively added elements are consistent with those already present, the puzzle should remain solvable; likewise, as with connecting new words to existing ones, added elements should meaningfully connect to existing elements. For example, other games with typically static clue lists could be modified to be reactively revealed, such as logic grid puzzles [16] by adding new elements to the puzzle that either clarify or confuse the present puzzle. This approach could also be applied to a *Sokoban*-like game, by having a "fog of war" mechanic that can change unrevealed areas to make levels easier or more challenging by adding new puzzle elements.

Because we used player personas, the quality of the word/puzzle clues was not an important part of the evaluation. The next step, though, is to assess the effectiveness of reactive puzzle construction with players, including whether the puzzle construction works for players. Before doing so, the dataset described in Section 3.1 will need to be improved to model surprisal based on the clue and the word [11]. For example, the word "from" is listed as the third-easiest word in our difficulty ordering, but one of the associated clues in the Clue-Instruct dataset is: "First season premiered on February 20, 2022." The other point to note is that the Clue-Instruct dataset has the additional context of category (e.g. literature, science, sports, etc.). The current version of the game does not provide that context, which could make the clues easier for actual participants.

Another area for future work is that the modifications to D_i were primarily aimed at making future words easier by reducing D_i by 10%. A similar logic could be applied to make future words more difficult by, for example, increasing D_i by 10%. This would allow the game to react by making the puzzle easier and harder, depending on the player.

It is important to note, though, that we have used reduced time as an evaluation of the approach working, but part of the goal with some crosswords is the challenge. Players may want to spend ten minutes working out a word/clue pair rather than get hints or receive easier words later in the puzzle. This method is not for them. This method is better applied in a learning context, where the goal is to keep the player in a state of flow [3].

References

[1] Adi Botea. 2007. Crossword grid composition with a hierarchical csp encoding. In *Proceeding of the 6th CP Workshop on Constraint Modelling and Reformulation, ModRef-07*.

[2] Adi Botea and Vadim Bulitko. 2025. Generating High-Score Crosswords Puzzles With Bi-Objective, Stochastic and Systematic Search. *ICGA Journal* 47, 1 (2025), 2–25.

[3] Mihaly Csikszentmihalyi. 1991. *Flow: The Psychology of Optimal Experience*. Harper Perennial, New York, NY.

[4] Sidney D'Mello and Art Graesser. 2014. Confusion and its dynamics during device comprehension with breakdown scenarios. *Acta psychologica* 151 (2014), 106–116.

[5] Nitin Gaikwad and Suresh Tankhiwale. 2012. Crossword puzzles: self-learning tool in pharmacology. *Perspectives on medical education* 1, 5 (2012), 237–248.

[6] Matthew L Ginsberg, Michael Frank, Michael P Halpin, and Mark C Torrance. 1990. Search Lessons Learned from Crossword Puzzles.. In *AAAI*, Vol. 90. 210–215.

[7] Miguel González-Duque, Rasmus Berg Palm, and Sebastian Risi. 2021. Fast game content adaptation through Bayesian-based player modelling. In *2021 IEEE Conference on Games (CoG)*. IEEE, 01–08.

[8] John Hale. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Second meeting of the north american chapter of the association for computational linguistics*.

[9] Sam Hughes. 2022. Absurdle. https://qntm.org/absurdle. Online game.

[10] Robin Hunicke. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*. 429–433.

[11] Tommaso Iaquinta, Asya Zanollo, Achille Fusco, Kamyar Zeinalipour, and Cristiano Chesi. 2025. Surprisal and Crossword Clues difficulty: Evaluating Linguistic Processing between LLMs and Humans. In *Proceedings of the Eleventh Italian Conference on Computational Linguistics (CLiC-it 2025)*. 512–525.

[12] Philip Edward Jaeger. 2000. *Cedar Grove*. Arcadia Publishing, Charleston, SC.

[13] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science* 331, 6014 (2011), 176–182.

[14] Wiwat Orawiwatnakul. 2013. Crossword puzzles as a learning tool for vocabulary development. *Electronic Journal of Research in Education Psychology* 11, 30 (2013), 413–428.

[15] Leonardo Rigutini, Michelangelo Diligenti, Marco Maggini, and Marco Gori. 2012. Automatic generation of crossword puzzles. *International Journal on Artificial Intelligence Tools* 21, 03 (2012), 1250014.

[16] Fiona Shyne, Kaylah Facey, and Seth Cooper. 2024. Procedurally puzzling: on algorithmic difficulty and player experience in QD-generated logic grid puzzles. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 20. 127–137.

[17] Sofia Eleni Spatharioti, Sara Wylie, and Seth Cooper. 2021. Exploring Q-Learning for Adaptive Difficulty in a Tile-based Image Labeling Game. In *2021 IEEE Conference on Games (CoG)*. 1–8. doi:10.1109/CoG52621.2021.9619125

[18] Martin Trenkmann. 2024. ngrams.dev: REST API for Google Books Ngram Data. https://ngrams.dev Accessed: 2026-04-12.

[19] Eric Wallace, Nicholas Tomlin, Albert Xu, Kevin Yang, Eshaan Pathak, Matthew Ginsberg, and Dan Klein. 2022. Automated crossword solving. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3073–3085.

[20] Josh Wardle. 2021. Wordle. https://www.nytimes.com/games/wordle/index.html. Online game.

[21] Arthur Wynne. 1913. Word-Cross Puzzle. *New York World* (21 Dec. 1913). First known published crossword puzzle.

[22] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* 1, 2 (2023), 1–124.

[23] Andrea Zugarini, Kamyar Zeinalipour, Surya Sai Kadali, Marco Maggini, Marco Gori, and Leonardo Rigutini. 2024. Clue-Instruct: Text-Based Clue Generation for Educational Crossword Puzzles. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. 3347–3356.