Generating Three-Star System Puzzles using Solution Enumeration Fitness for a Lattice Protein Folding Game

Yaejie Kwon[1], Fiona Shyne[2] and Seth Cooper[2]

[1]Swarthmore College
[2]Northeastern University

Abstract

Many puzzle games incorporate multi-star systems that provide different rewards for player performance on each puzzle, with better performance earning more stars. Such puzzles require distinct solutions of increasing difficulty; however, most work in generating puzzles considers just a single solution. In this work, we explore an approach to generating puzzles that have many distinct solutions and could thus lend themselves to being used in a game with a multi-star (in this case, three-star) rating system. Our approach uses a genetic algorithm to search over possible puzzles, and incorporates a fitness function that exhaustively enumerates all possible solutions, rewarding solutions with more distinct solutions. We apply the approach to a game based on a simple lattice protein folding, where puzzles are determined by the a sequence of amino acids, and fitness enumerates all possible conformations with different numbers of contacts.

Keywords

procedural content generation, star rating, enumeration

1. Introduction

It is not uncommon for puzzles in games to have multiple ways to be solved, with some being considered better than others. This can take form in getting a higher score, making fewer moves, using less time, or taking less common approaches (for example: Angry Birds, Dragon Box, Where's my Water, and Cut the Rope). In such games, players can be rewarded for better solutions by what we refer to as a *three-star system*, where a player is given up to three stars for their performance on a puzzle.

A critical aspect of puzzle generation is that a puzzle must be solvable; that is, it must be possible to reach the solution of the puzzle. There are many approaches to procedural content generation [1] for creating puzzles for games [2]. However, less explored is the generation of puzzles with multiple distinct solutions that would lend themselves to a three-star system puzzle game.

Therefore, in this work, we developed an approach to generating three-star puzzles that incorporates a search over possible puzzles with an exhaustive enumeration of their solutions. More specifically, we use a genetic algorithm to search over puzzles, in which the fitness function enumerates the solutions for that puzzle and prefers those that have a larger number of distinct possible scores. Such puzzles may lend themselves better to gradually rewarding players with increasing stars as they find progressively better solutions by associating each of the three stars with a score threshold. We apply this approach to a prototype puzzle game, *Foldit Lattice*, built around the 2D lattice protein folding problem [3, 4]. The concept of *Foldit Lattice* is to be a more lightweight and "casual" introduction to the *Foldit* citizen science biochemistry game [5].

Our findings show that this approach can be used to generate puzzles with a range of different scores, and that the genetic algorithm can effectively evolve an initial population of puzzles that contain relatively few high-fitnesses into a larger set of top-fitness puzzles.

Joint AIIDE Workshop on Experimental Artificial Intelligence in Games and Intelligent Narrative Technologies, November 10-11, 2025, Edmonton, Canada.

ykwon1@swarthmore.edu (Y. Kwon); shyne.f@northeastern.edu (F. Shyne); se.cooper@northeastern.edu (S. Cooper)
0009-0009-6381-5905 (Y. Kwon); 0009-0009-9222-5242 (F. Shyne); 0000-0003-4504-0877 (S. Cooper)

2. Related Work

2.1. Procedural Puzzle Generation

PCG for puzzles remains under-explored, as puzzle generation must ensure playability, appropriate difficulty, and other criteria are often game-specific and more complex than other PCG domains [2]. The creation of puzzles, compared to that of levels or other content, requires more organization as you must make sure that they are solvable, non-trivial, and have the desired difficulty [1]. Puzzles often need to be evaluated with fitness functions that measure properties such as uniqueness, optimality, or number of solutions. For this reason, complete coverage and strong evaluation are established with exhaustive searches that find all possible configurations for a puzzle. Many procedural puzzle generation methods typically focus on generating puzzles with one correct solution, rather than designing those that support multiple distinct solutions.

There have also been uses of genetic algorithms applied to procedural puzzle generation. For instance, a variant of FI-2pop was used to generate puzzles that can be described as a sequence of actions. The optimization functions in the algorithm attempt to adjust the difficulty based on the player and utilize a simple player model to control this [6]. Similarly, genetic algorithms have been applied to construct grid-based logic puzzles, where fitness was determined by moves done by an automatic solver, and difficulty was calculated according to the types of moves the solver would use [7]. Another approach [8] used a GA to generate clues for a children's game of placing tiles in a row, measuring difficulty by the number of possible solutions, and used AI-generated narratives from Chat-GPT for each of the clues.

2.2. Exhaustive Search

The approach to our current game design requires looking at all solutions for a puzzle, building upon the foundation of exhaustive and semi-exhaustive procedural content generation (EPCG) [9] and large-scale search of game content [10]. Although this method has yet to be largely studied, it allows for a systematic way of identifying all possible content in a given search space. When applying this to puzzle-like games, EPCG contains features to map between a puzzle state and the rank in the context of its design. This also allows selecting and evaluating content based on desirable properties, such as the difficulty of the level. Our approach would extend these concepts to a more scientific puzzle and connect them to multi-solution generation for three-star systems. Recently, EPCG approaches have been used in the design process of novel puzzle games, displaying systematic search methods to generate and evaluate puzzle content [11].

2.3. Three-Star System

A three-star system is one that is commonly seen in these games with puzzle-like aspects, and has been found to help increase user engagement and adoption [12]. Research has found that this progression allowed for a positive response from players as they tended to use fewer moves because they carried out each one more thoughtfully, and replayed levels more often to optimize the level score. Despite this behavioral impact, it was also found that player retention did not improve due to usually having only one inherent solution for the level. However, because our prototype game has the main purpose of being an introductory game, its feature of having multi-solution puzzles at each threshold may allow for a stronger retention rate compared to those that were examined in this research. However, the impact, replay rate, and rewards are not entirely clear [13].

2.4. Science-Related and Crowdsourcing Games

Many crowdsourced, citizen-science games engage the public with real scientific data through puzzle-like mechanics. These include Foldit [5] or EteRNA [14], which involve concepts such as protein-folding and RNA molecule design, where solutions by players can be integrated into real experiments. Other

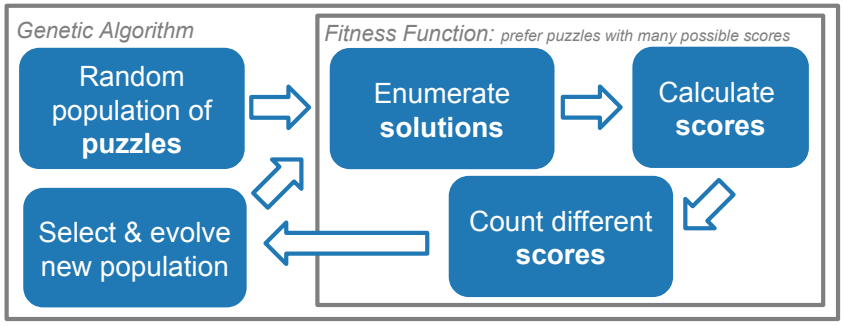


Figure 1: Flowchart of the general approach to finding puzzles with many possible solutions. A genetic algorithm is used to evolve a population of **puzzles**. The fitness function evaluates a puzzle by enumerating all **solutions** for a puzzle, calculating their **scores**, and counting the number of different scores found; more different scores have higher fitness. The fitness values are used to select and evolve the new population, and the process repeats.

puzzle-based games, such as Phylo [15], Borderlands Science, and Fraxinus [16], all integrate other approaches to science games that allow for a much simpler understanding of these scientific concepts, all whilst allowing the users to contribute to a larger cause that may deal with things such as genomes. Despite the fact that these games help make scientific concepts more approachable through games, many of them lack accessibility and long-term engagement due to their complex mechanics.

There are also other, more casual crowdsourced games useful for collecting information. For example, ESP Game [17] was used to produce metadata made by humans to support image creation, BeFaced [18] assisted in face perception studies, and Cropland Capture [19] found training data for machine learning in agriculture through its mobile game, where users labeled satellite images of farmland. This shows that it is possible to construct meaningful puzzles that are approachable in design, while having layered difficulty.

3. General Approach

Genetic algorithms are a search method inspired by natural selection. We used this method to evolve optimal solutions that produce a wide range of solution scores, with the goal of providing more options for 3-star thresholds and ideally more engaging level design.

A flowchart giving an overview of the process is shown in Figure 1. Our genetic algorithm begins with a random population of puzzles. Each puzzle is represented as a string of elements, in which each element corresponds to a different character based on the puzzle description. This population is then input into a fitness function, which is used to numerically evaluate the quality of each individual in the population. In this project, puzzles are given preference over when there are a variety of different scores, allowing more possibilities to reward the players with stars. The fitness function starts by exhaustively iterating through all possible solutions of the puzzle. The fitness score is found by calculating the total number of distinct scores the puzzle can produce across its valid solutions (i.e., higher diversity of scores indicates a better level).

Based on the fitness score, parents of puzzles are selected randomly with a weighted chance proportional to the score. To these parents, mutation and crossover functions are applied to generate offspring puzzles. For crossover, a one-point approach is used, where a single random point is chosen, and all elements after that point are swapped between the two parents. Mutation randomly changes individual elements of a puzzle, introducing additional diversity. These offspring are then inserted back into the population, and the cycle repeats until the set number of generations is complete. As a result, the algorithm yields puzzles with the highest fitness.

The types of application that this algorithm could be applied to consist of those in which it is possible to enumerate and search through all permutations of a puzzle's solution, which should have a range of possible scores. Thus, this framework can be further implemented into games, such as Sokobond [20].

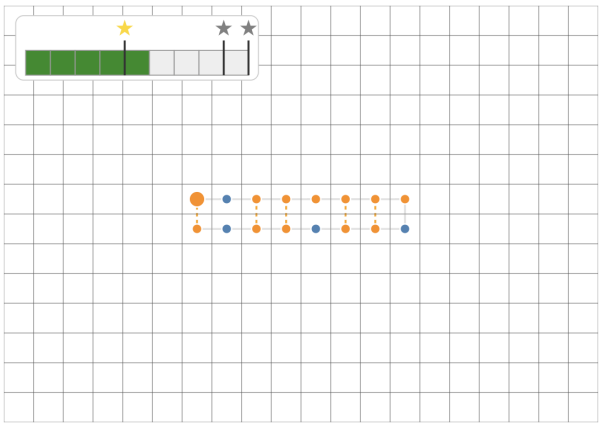


Figure 2: Screenshot of Foldit Lattice. The current conformation of the protein is in the center; H nodes are orange and P nodes are blue. HH bonds are orange dotted lines. The player's score, which is the number of HH bonds, is shown in the top-left, along with the score thresholds for stars.

4. Foldit Lattice Game Description

To test out our approach, we applied it to a prototype game, *Foldit Lattice*. *Foldit Lattice* was inspired by its larger, more complex counterpart, Foldit. It is based on the lattice protein folding problem and uses the 2D hydrophobic-hydrophilic model [3]. A screenshot is shown in 2.

In this game, players are provided with a protein chain consisting of hydrophobic (orange) and hydrophilic (blue) nodes. Based on the length and the order of this sequence of hydrophobic and hydrophilic nodes, the connected chain can be manipulated by having orthogonal moves (90° or 180°) between each node. An arrangement of a puzzle's nodes (i.e., a solution) is referred to as a *conformation*. Players can freely drag the nodes with the mouse using a mass-spring model; the nodes are also pulled to the center of cells in the visualized grid to keep them roughly orthogonal. The player's goal is to maximize the amount of HH (hydrophobic-hydrophobic) contacts. These contacts happen when two hydrophobic nodes are adjacent to each other on the grid, but not next to each other based on the order of the chain (as seen through the orange dotted lines in Figure 2). In the case of Foldit Lattice, the solution of a puzzle is a protein conformation–essentially the directions from a node to the following one–and the score is the total number of HH contacts.

4.1. Applying Approach to Foldit Lattice

In the case of Foldit Lattice, to create an optimal puzzle for the game, we began with a random population of hydrophobic (H) and hydrophilic (P) string sequences of fixed length. Each puzzle is represented as a linear sequence of the H and P elements, which can be thought of as a 1D array. We then formulated a fitness function to enumerate all valid conformations that avoid self-intersection for each sequence. Its representation of each node on the grid would be through a string of directions (up, down, left, right), avoiding symmetric redundancies. For performance, we implemented enumeration in C, based on the `harmslab/latticeproteins` Python package [21, 22, 23] and a tutorial [24]. This uses a recursive depth-first search with backtracking, which also handles symmetry considering rotations and reflections. For each of the conformations, the HH-contact score is calculated, and the fitness score is given based on how many distinct scores the puzzles can have. With this fitness score, sequences are randomly selected again with a weighted chance based on their respective scores. Mutations randomly change an element from H to P or vice versa, and crossover generates offspring by exchanging subsequences between two parent sequences. This process is repeated until the sequences with the best fitness are found.

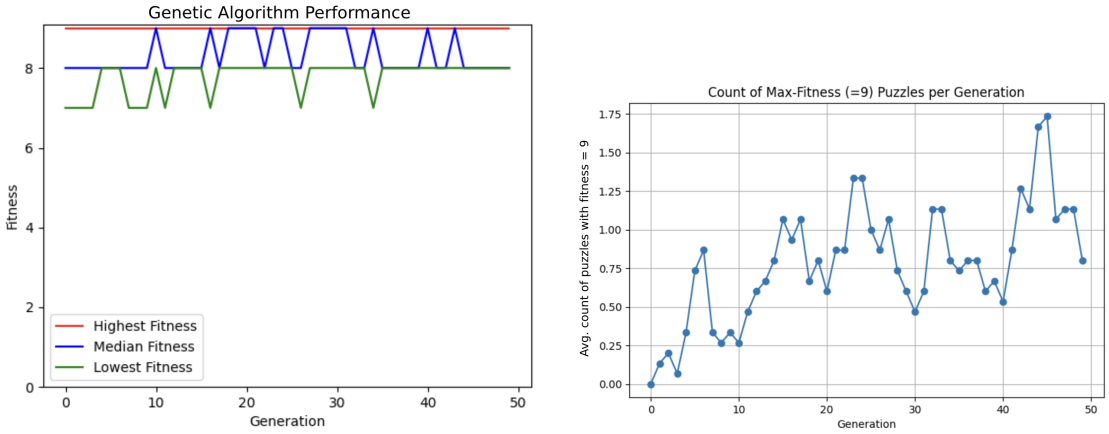
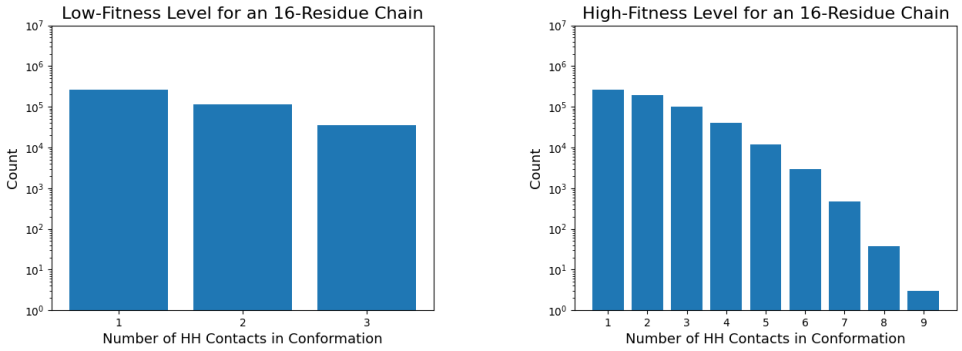


Figure 3: (left) Plot showing the range of highest, median, and lowest fitness values in the population across 50 generations for puzzles of length 16. As the maximum fitnesses are consistently high, it indicates that some high-performing puzzles were found early on in the generations. (right) Plot of the average number of puzzles with the maximum fitness (9) for length 16 sequences in each generation across five trials. The upward trend shows that the genetic algorithm increasingly finds high-fitnesses over time.



Low-Fitness Puzzle High-Fitness Puzzle

Figure 4: Histograms of the number of total conformations corresponding to each possible HH-contact score in low- and high-fitness puzzles of length 16. The low-fitness puzzle (left) has only three bottom-range scores, while the high-fitness puzzle (right) has a broader range of possible scores (up to 9). This higher diversity in achievable scores suggests more potential for difficulty and star thresholds in high-fitness puzzles.

5. Evaluation

The goal of this evaluation is to examine the performance of the genetic algorithm in generating puzzles with diverse fitness scores.

For all experiments, we used a population size of 30, 50 generations, single-point crossover, a point-wise mutation at a rate of 0.1 per element, and 1 elite (the highest-fitness puzzle in each generation is carried over unchanged). For overall GA performance, 25 trials were averaged, and for max fitness count, 15 trials were averaged.

We focused on puzzles of length 16, but ran puzzles of smaller sizes to get rough timing information. Experiments were run on a MacBook Pro (Apple M2, 8-core CPU with 8 GB memory), and we parallelized the evaluation of puzzle solutions in our code to reduce computation time. For puzzles with sequence length 16, the total time was approximately 2.7 hours (162 minutes), with an average of 31.5 minutes per sequence. Each trial took an average of 390 seconds (6.5 minutes), and each generation took approximately 7.8 seconds. For puzzles with sequence length 8, the total time was approximately 54 minutes, with an average of 10.9 minutes per sequence. Each trial took an average of 131 seconds (2.2

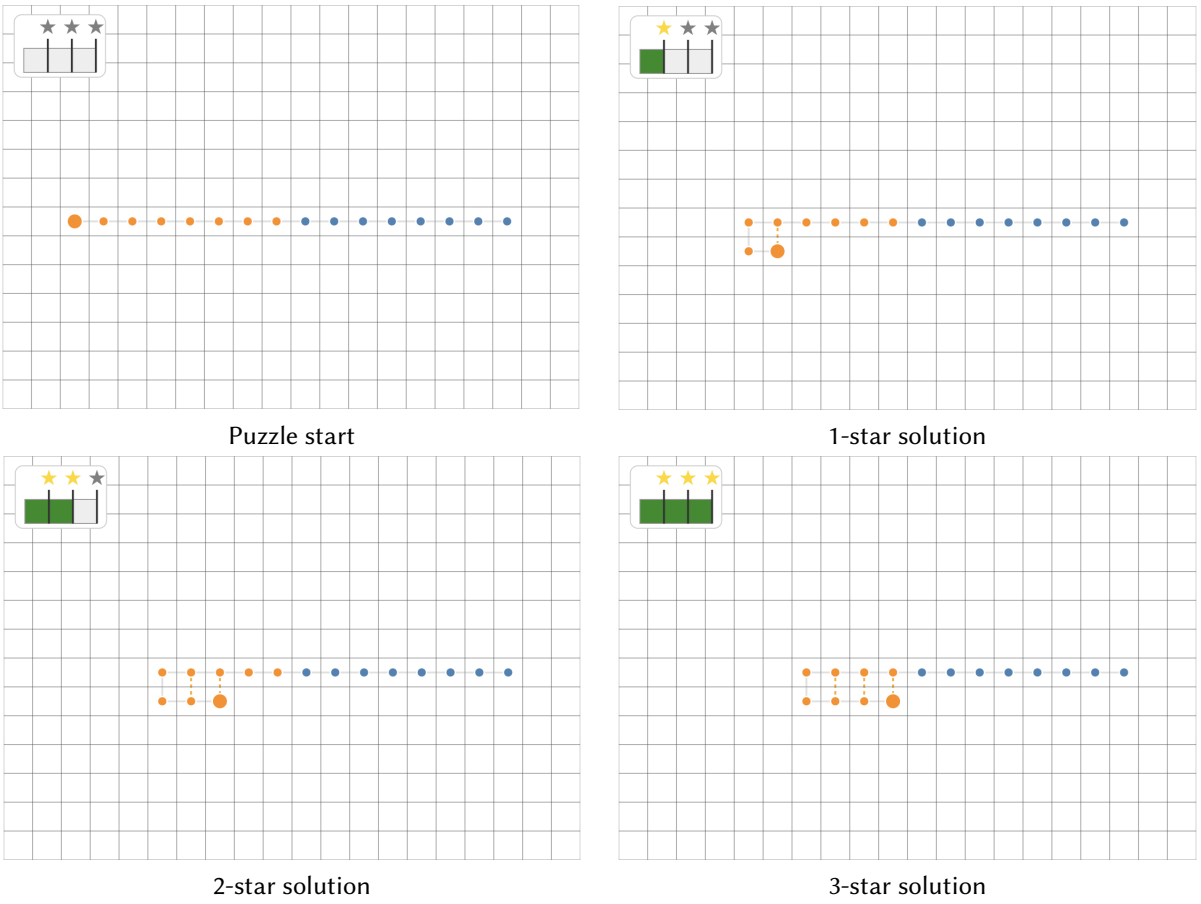


Figure 5: Example (manually created) low-fitness puzzle. Although this puzzle does have up to three stars, the possible solutions are limited and visually similar to one another. The simple construction of the puzzle results in lower variation in achievable scores and potentially less player engagement compared to the high-fitness puzzle.

minutes), and each generation took approximately 2.6 seconds. Finally, for puzzles with sequence length 4, the total computation time was around 47 minutes, with an average of 9.1 minutes per sequence. Each trial took about 109 seconds (1.82 minutes), and each generation took roughly 2.2 seconds.

Based on the genetic algorithm performance plot, we observed a consistent range of high fitness scores throughout the generations.

Figure 3 shows trajectories of the population of the genetic algorithm over the generations. On the left, we see that in fact the first (randomly-generated) population performs quite well, finding solutions with the maximum score found overall. However, on the right, we see that the number of puzzles in the population with that maximum score increases roughly over the generations. This demonstrates that the population is improving over time; finding more of these high-scoring puzzles would be useful for a game, as there are more such puzzles to choose from.

Figure 4 represents levels of the Foldit Lattice, which consists of low and high-fitness 16-length sequences. When mapping this onto a histogram, we can visualize what does and does not classify as an ideal level. For example, the low-fitness histogram only has a total of 3 different scores achievable with a similar number of configurations applicable to each score value. On the other hand, the high-fitness histogram provides a much larger range of scores, with a downwards sloping trend where there are very few possible solutions for the highest score, 9, and many more for the lower scores. This distribution could allow for a more stimulating experience for the player as it promotes critical thinking.

Using the distribution seen on these graphs, we can assign the thresholds for the 3-star system for each level. For the Figure 6 puzzle that represents the high-fitness sequence in Figure 4, because the maximum achievable score found is 9, the star threshold is divided by 0-3 as zero stars, 4-7 contacts as

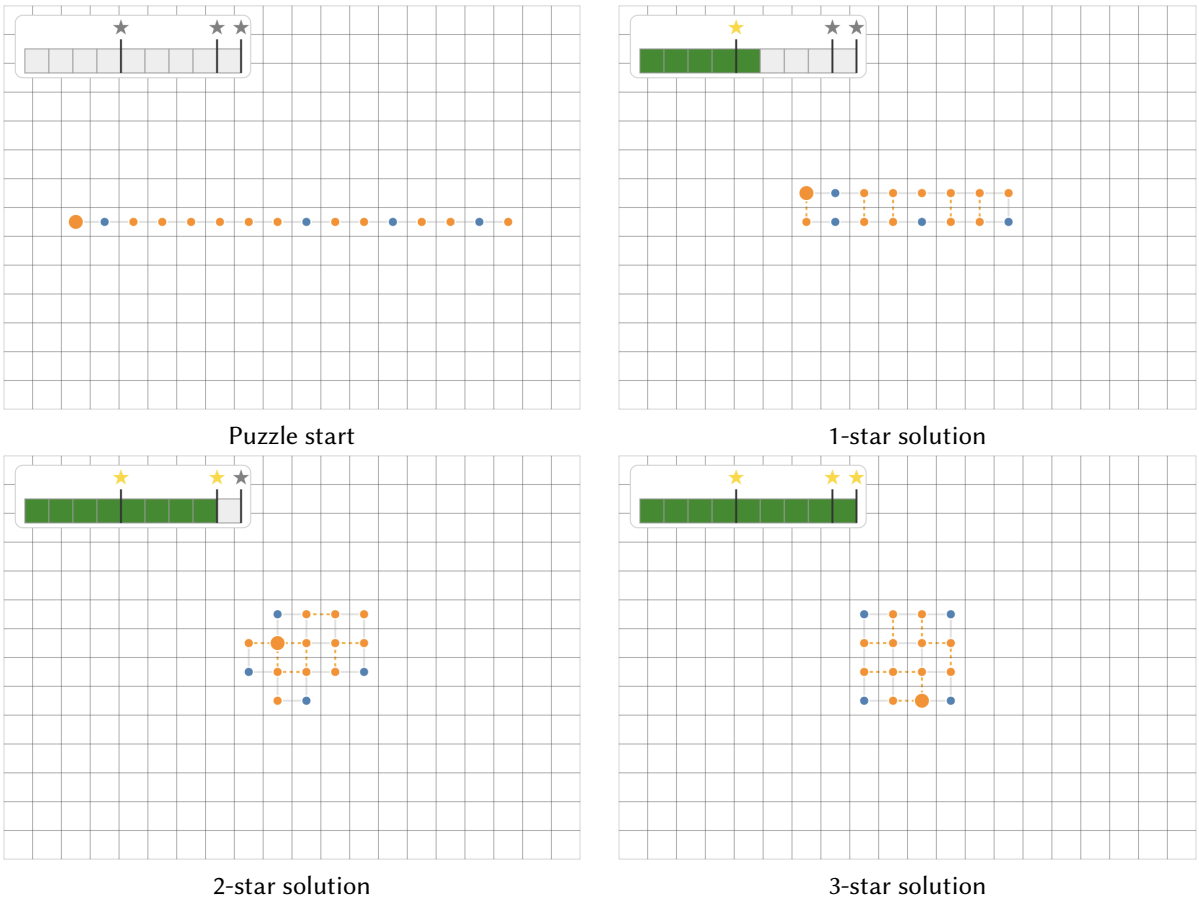


Figure 6: Example high-fitness puzzle with possible solutions for each star level. As shown, the solutions for this level are more involved. The 2 and 3-star solutions are notably more interwoven than the lower-star solutions, requiring players to make intricate, thought-inducing folds to maximize the number of hydrophobic–hydrophobic (HH) contacts.

one star, 8 contacts as two stars, and 9 contacts as three stars. On the other hand, when determining the score structure for the low fitness sequence shown in Figure 5, a star is obtained with each additional HH-contact, as there are only a maximum of three contacts possible. In this puzzle, we can also observe that the difficulty and subjective visual interest are also low because it is very straightforward to get each contact with the hydrophilic residues positioned right next to each other.

6. Conclusion

In this work, we were able to successfully generate levels for puzzles with multiple solutions that fit the three-star thresholds. Using a genetic algorithm approach, the number of top-scoring puzzles in the population increased over the generations.

In the future, we would like to adopt a quality-diversity search in place of the genetic algorithm. This approach would provide a variety of puzzles with competitive solutions in the search space, rather than just a singular “best” puzzle.

We also intend to study the spacing of stars among the score thresholds. There are various methods to do this, such as choosing specific percentiles, deciding based on the model of the graph, or by the number of total differing scores. In this work, we only looked at the number of different scores in the fitness function; however, the shape of the score histogram may also be interesting to consider (e.g., the relative proportions of each score). There may also be technical scalability issues for larger puzzles whose solutions are increasingly difficult to enumerate.

Specifically, in the context of Foldit Lattice, we would like to incorporate more complex functionality

of lattice protein models. This would include factors such as working with a 3D structure instead of a 2D one. It would also involve parts of the genetic algorithm so that it integrates additional attributes to the hydrophobic-hydrophilic model, such as how the score is calculated, rather than simply counting the number of HH contacts. We are also considering conducting a user study to help evaluate the differences in engagement from the user between low and high fitness puzzles.

Lastly, having only applied this to Foldit Lattice, we propose an extension of this method to other existing games that have the required characteristics of project description, variety of solutions, and score.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2244288 and the Rosetta Commons REU Program. We would also like to thank Jaden Rodriguez for his contributions.

Declaration on Generative AI

During the preparation of this work, the author(s) used Grammarly in order to: Grammar and spelling check. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] N. Shaker, J. Togelius, M. J. Nelson, Procedural Content Generation in Games, Springer International Publishing, 2016.

[2] B. De Kegel, M. Haahr, Procedural puzzle generation: a survey, IEEE Transactions on Games 12 (2020) 21–40. doi:10.1109/TG.2019.2917792.

[3] K. F. Lau, K. A. Dill, A lattice statistical mechanics model of the conformational and sequence spaces of proteins, Macromolecules 22 (1989) 3986–3997. URL: https://doi.org/10.1021/ma00200a030. doi:10.1021/ma00200a030, publisher: American Chemical Society.

[4] P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, M. Yannakakis, On the complexity of protein folding, Journal of Computational Biology 5 (1998) 423–465. URL: https://www.liebertpub.com/doi/10.1089/cmb.1998.5.423. doi:10.1089/cmb.1998.5.423, publisher: Mary Ann Liebert, Inc., publishers.

[5] S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popović, Foldit Players, Predicting protein structures with a multiplayer online game, Nature 466 (2010) 756–760. URL: http://dx.doi.org/10.1038/nature09304. doi:10.1038/nature09304.

[6] M. Scirea, Adaptive puzzle generation for computational thinking, in: X. Fang (Ed.), HCI in Games. HCII 2020, volume 12211 of *Lecture Notes in Computer Science*, Springer, Cham, 2020, pp. 429–441. doi:10.1007/978-3-030-50164-8_35.

[7] D. Oranchak, Evolutionary algorithm for generation of entertaining shinro logic puzzles, in: Applications of Evolutionary Computation. EvoApplications 2010, volume 6024 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 2010, pp. 208–217. doi:10.1007/978-3-642-12239-2_19.

[8] T. Volden, D. Grbic, P. Burelli, Procedurally generating rules to adapt difficulty for narrative puzzle games, in: 2023 IEEE Conference on Games (CoG), IEEE, 2023, p. 1––4. URL: http://dx.doi.org/10.1109/CoG57401.2023.10333251. doi:10.1109/cog57401.2023.10333251.

[9] N. R. Sturtevant, M. J. Ota, Exhaustive and semi-exhaustive procedural content generation, AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment; Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference (2018).

[10] N. R. Sturtevant, An argument for large-scale breadth-first search for game design and content generation via a case study of Fling!, in: Proceedings of the AIIDE Workshop on Artificial Intelligence in the Game Design Process, 2013, pp. 28–33. URL: http://web.cs.du.edu/~sturtevant/papers/BFS-design.pdf, precomputation testing debugging.

[11] Y. Mahmoud, N. R. Sturtevant, Using epcg for designing a hexagon tangram puzzle, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, volume 20, 2024, pp. 199–207. doi:10.1609/aiide.v20i1.31880.

[12] J. Gaston, S. Cooper, To three or not to three: improving human computation game onboarding with a three-star system, in: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, CHI '17, ACM, New York, NY, USA, 2017, pp. 5034–5039. URL: http://doi.acm.org/10.1145/3025453.3025997. doi:10.1145/3025453.3025997.

[13] Y. Long, V. Aleven, Gamification of joint student/system control over problem selection in a linear equation tutor, in: Intelligent Tutoring Systems, Lecture Notes in Computer Science, Springer, Cham, 2014, pp. 378–387. URL: https://link.springer.com/chapter/10.1007/978-3-319-07221-0_47. doi:10.1007/978-3-319-07221-0_47.

[14] J. Lee, W. Kladwang, M. Lee, D. Cantu, M. Azizyan, H. Kim, A. Limpaecher, S. Yoon, A. Treuille, R. Das, EteRNA Participants, RNA design rules from a massive open laboratory, Proceedings of the National Academy of Sciences 111 (2014) 2122–2127. URL: http://dx.doi.org/10.1073/pnas.1313039111. doi:10.1073/pnas.1313039111.

[15] A. Kawrykow, G. Roumanis, A. Kam, D. Kwak, C. Leung, C. Wu, E. Zarour, L. Sarmenta, M. Blanchette, J. Waldispühl, Phylo Players, Phylo: a citizen science approach for improving multiple sequence alignment, PLOS ONE 7 (2012) e31362. doi:10.1371/journal.pone.0031362.

[16] G. Rallapalli, Fraxinus Players, D. G. Saunders, K. Yoshida, A. Edwards, C. A. Lugo, S. Collin, B. Clavijo, M. Corpas, D. Swarbreck, M. Clark, J. A. Downie, S. Kamoun, Team Cooper, D. MacLean, Lessons from Fraxinus, a crowd-sourced citizen science game in genomics, eLife 4 (2015) e07460. doi:10.7554/eLife.07460.

[17] L. von Ahn, L. Dabbish, Labeling images with a computer game, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, Vienna, Austria, 2004, pp. 319–326. doi:10.1145/985692.985733.

[18] C. T. Tan, H. Sapkota, D. Rosser, BeFaced: a casual game to crowdsource facial expressions in the wild, in: CHI '14 Extended Abstracts on Human Factors in Computing Systems, CHI EA '14, ACM, New York, NY, USA, 2014, pp. 491–494. URL: http://doi.acm.org/10.1145/2559206.2574773. doi:10.1145/2559206.2574773.

[19] T. Sturn, M. Wimmer, C. Salk, C. Perger, L. See, S. Fritz, Cropland Capture – a game for improving global cropland maps, in: Proceedings of the 10th International Conference on the Foundations of Digital Games, 2015.

[20] Thinking Rabbit, Sokoban, 1982. Game.

[21] Z. Sailer, J. Bloom, M. Harms, Latticeproteins, https://github.com/harmslab/latticeproteins/, 2013.

[22] J. D. Bloom, C. O. Wilke, F. H. Arnold, C. Adami, Stability and the evolvability of function in a model protein, Biophysical Journal 86 (2004) 2758–2764. doi:10.1016/S0006-3495(04)74329-5.

[23] J. D. Bloom, S. T. Labthavikul, C. R. Otey, F. H. Arnold, Protein stability promotes evolvability, Proceedings of the National Academy of Sciences of the United States of America 103 (2006) 5869–5874. doi:10.1073/pnas.0510098103.

[24] S. Will, Lattice models: The simplest protein model, https://math.mit.edu/classes/18.417/Slides/HP-protein-prediction.pdf, 2011.