Evolving Romanian Crossword Puzzles with Deep Learning and Heuristic Search

Vadim Bulitko
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
bulitko@ualberta.ca

Adi Botea
Eaton
adibotea@eaton.com

***Abstract*—Crossword puzzles are a challenging game of skill in which humans have been competing for decades. Recently a heuristic-search-based Artificial Intelligence (AI) solver for Romanian crossword puzzles achieved competition-level scores. In this work in progress we tackle procedural content generation of crossword puzzles. Using genetic algorithms we evolve crossword puzzle instances on which the AI solver can achieve a high score. Since the solver takes a substantial time to solve each instance we first train a deep neural network to predict the solution score the solver would achieve. We then run the evolution of crossword puzzles with the network as the fast proxy fitness function. We show that doing so is an effective way of procedurally generating numerous crossword puzzles.**

***Index Terms*—genetic algorithms, proxy fitness, deep learning, procedural content generation, crossword puzzles**

I. Introduction

Crossword puzzles are a popular entertainment option in many languages. Thematic puzzles, where part of the words in the solution belong to a common theme, are particularly interesting. Besides entertainment, thematic puzzles can have an educational value, presenting interesting information within the theme chosen, through the clues and through the solutions. The quality of a thematic crossword puzzle can increase when the thematic contents is higher. Thus one can define *thematic score* as the total length of thematic words used in filling out a crossword puzzle.

Solving a crossword puzzle is a computational problem: given a grid size, a pattern of black cells and a list of valid words, the task is to find a way to fill the grid with words. With no score defined, many valid solutions can be equally acceptable. Defining a scoring function, such as the thematic score, converts this into an optimization problem. In practice, finding a high-score solution can be more challenging than finding a valid solution. However, the decision problems associated with each of these problems (i.e., with and without the optimization component) have a similar theoretical complexity, namely NP-complete [1], [2].

Interestingly, the ability to rank crossword grids, thanks to the availability of a thematic score, has led to competitions between human contestants. The Romanian Crossword Competition is a decades-old competition organized by Rebus, a Romanian publication dedicated to crossword puzzles and other mind games. Humans have achieved an impressive level of expertise, beyond the reach of current AI algorithms.

Recent work [2] presented a search-based approach to solving Romanian Crossword Competition puzzles. The proposed program WOMBAT achieved champion-level scores in filling a grid with words from generic and thematic dictionaries. In other words, WOMBAT addresses only a sub-problem of the Romanian Crossword Competition problem by taking an unfilled grid with black cells as an input.

In this paper we tackle the remaining problem: creating patterns of black cells on a square grid. The resulting grids can then be given to WOMBAT as an input and filled with words. To solve this problem we use genetic algorithms to evolve crossword puzzles with high scores (as computed by WOMBAT). Since WOMBAT itself is too slow to be used as a fitness function during the evolution we first train a deep neural network to predict WOMBAT score. We then use the trained network as a proxy fitness function in the evolution. To the best of our knowledge, this is the first published account of procedurally generating Romanian Competition Crossword puzzle instances.

II. Problem Formulation

An instance of a *Romanian Competition Crossword puzzle* is a tuple $P = (G, D_g, D_t)$ where G is a 13×13 binary grid with at most 26 black cells (the rest being empty) and D_g and D_t are general and thematic dictionaries. The grid G must satisfy the following constraints: (C1) no two black cells can be adjacent cardinally, (C2) using only cardinal moves one must be able to traverse G from any empty cell to any other empty cell and (C3) it should not be possible to change an empty cell to a black cell and violate constraint C2 unless the isolated areas created by the change contain one empty cell.

A crossword grid G defines *word slots*: maximally contiguous vertical and horizontal sequences of empty cells. A *solution* to the puzzle instance P is a mapping of horizontal and vertical word slots in G to words from the dictionaries D_g and D_t. Slots of length one can be filled with any letter. Slots of length two can be filled with any combination of two letters regardless of D_g and D_t. Neither such two-letter combinations nor dictionary words can repeat in a solution.

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

The *solution score* is the sum of lengths of all thematic words (i.e., words from D_t) in the solution. Note that one can score a *full solution* which leaves no empty cells in G or a *partial solution* which has some unfilled empty cells.

The **problem we tackle in this paper** is, given dictionaries D_g, D_t, to generate a grid G that allows for a high-score solution. We would like the grid generation to be procedural and efficiently produce numerous grids with high solution scores. Additionally we prefer grids that allow for full solutions.

III. Related Work

Procedural content generation is an active field in games with various types of game content successfully generated in both research [3]–[8] and in the field [9]–[12]. In particular, procedural puzzle generation is surveyed by De Kegel and Haahr [13]. Douglas, et al. [14] use genetic algorithms to generate crosswords puzzles. However, we focus on the optimization version of the problem, aiming at generating grids with a high score.

Published work on crosswords can be classified into three categories: solving a puzzle (i.e., filling the words in an existing grid, based on an existing set of clues [15]), generating a crossword grid with no notion of a score [16], [17] and generating a crossword grid where a higher score is preferred (which is the problem we tackle in this paper). To our knowledge, published work in the third category pertains to the Romanian Crossword Competition [2], [18], [19].

IV. Our Approach

To procedurally generate high-score crossword puzzles one needs to be able to (i) compute a solution score of such puzzles and (ii) effectively search a space of crossword puzzles.

Computing puzzle scores. Given a crossword puzzle (i.e., a grid and dictionaries) we first need to compute a high-quality solution in order to calculate the solution score. To compute a solution to a crossword puzzle instance one can engage a skillful human but doing so is not tractable for many instances. An AI crossword puzzle solver, WOMBAT, is able to produce competition-level solutions with scores comparable to those of human solutions [2]. Thus in our approach we use WOMBAT to solve crossword puzzles and calculate their scores.

Searching the space of grids. Note that not only valid grids in Romanian crossword puzzle are subject to constraints C1 – C3 (Section II) but also their word slots have to be of suitable lengths as long slots reduce the chance of filling them with score-accumulating thematic words.

To search the space of grids we use a basic genetic algorithm as follows. At each generation a population of N crossword puzzle instances is evaluated using a fitness function f (discussed below). The bottom M members of the population are discarded and the remaining $N - M$ instances are transferred to the next generation. To build the population back up to N instances we select a random parent from the $N - M$ kept instances and mutate it to produce a offspring. We repeat this asexual reproduction M times per generation. Throughout the evolution we maintain a running best: the best fitness grid encountered. The evolution runs for pre-specified number of generations.

Fitness function. WOMBAT takes minutes to solve a single puzzle instance and is thus too slow to be used as the fitness function. Thus we use a *proxy fitness* function. To do so we first run WOMBAT on randomly generated puzzle instances and compute their scores. The resulting pairs of grids and their scores are then used to train a deep neural network (ANN). The fitness function is then $f(G) = \alpha \cdot l_G - s_G$ where G is the grid and s_G is its ANN-predicted score; l_G is the length of the maximum word slot in G and α is a constant. As lower values of l_G tend to improve WOMBAT score, lower f values are preferred in evolution. We pick α to be larger than scores predicted by the ANN. Thus our fitness function combines two objectives: lower maximum word-slot lengths and higher predicted solution scores. In practice l_G provides a coarse guidance since many grids in a population will share the same value of l_G. The ANN-predicted, real-valued score then imposes a preference among them.

V. Empirical Evaluation

We compared three methods of procedurally generating crossword puzzle grids. Our implementation used a mixture of MATLAB and C++ code. Readers interested in our code and data are welcome to contact us. Throughout the experiments both dictionaries remained fixed.

Random grid generation. We started by generating random combinations of 26 black cells out of 13×13 possible cells. Each such combination defines a crossword grid. We discarded all duplicate grids as well as the grids which did not satisfy the constraints C1 through C3. We then ran WOMBAT on the remaining 1046 grids. Specifically, we ran WOMBAT in a best-first search mode, guided by the evaluation function $e(s) = p(s) - w \cdot \ell(s)$, where $p(s)$ is the score of state s achieved so far, $w = 1.5$ and $\ell(s)$ is the sterile load of s (the sum of the lengths of all slots committed to be instantiated with a regular (non-thematic) word). The target score was set to 140 points. A partial state was pruned if its current score p plus an optimistic estimation of the score that can be achieved in the rest of the grid was lower than the target score. The time limit was set to 20 minutes and the memory to maximum 2GB per instance on Compute Canada.

WOMBAT did not find a full solution for any of the 1046 random grids. The average partial solution score was 59.22 (Table I). The grids, their transpositions* and WOMBAT scores formed a data set of 2092 (grid, score) pairs.

Evolution with the max-word slot length. We then ran evolution with the maximum word-slot length (mWSL, denoted by l_G in Section IV) as the sole fitness function (i.e., $f = l_G$). Population size was set to $N = 1000$ and the initial population was randomly drawn without repetition from the 1046 grids described earlier. On each generation the $M = 900$ least fit grids were discarded. The remaining $N - M = 100$

*A transposition of a grid is transposition of its binary matrix. It has the same score since transposing merely swaps vertical and horizontal word slots.

grids were transferred to the next generation. These 100 grids also became parents and gave birth to $M = 900$ offsprings. Each offspring was formed by mutating one of the 100 parents chosen at random. The number of mutations per reproduction was $m = \lceil 0.1 + |e| \rceil$ where e is drawn from an exponential distribution with the mean parameter $\mu = 5$. Each mutation was a shift of a randomly chosen black cell to a randomly chosen empty neighboring cell. After the m mutations the resulting grid is checked against the constraints C1 through C3. An offspring that violates any constraint is discarded and another offspring is created by mutating the parent.

The evolution ran for 100 generations (Figure 1, top). We then removed duplicates from the final population and ran WOMBAT on the remaining 942 grids. WOMBAT successfully scored 933 of them[†], finding no full solutions. Adding the transpositions, we formed a set of 1866 (grid, score) pairs. The average partial score was 95.7 (Table I).

Training the ANN. We randomly partitioned the 2092 random grids and their transpositions into 1992 grids for training and 100 grids for testing. Then a deep neural network was trained on random 80% of the 1992 training grids with the other 20% being used for validation.

The input of the ANN was a $13 \times 13 \times 1$ matrix with black cells represented by $+0.5$ and empty cells represented by -0.5. The input layer was followed by five successive 2D-convolution layers with batch-normalization and ReLU layers in between them. The last convolution layer was connected via a dropout layer (probability of 0.2) to six successive fully connected layers. We put batch-normalization, ReLU and dropout layers in between the fully connected layers. The last fully connected layer had a single neuron used as the regression output (i.e., predicted solution score) of the ANN.

Training ran for fewer than 100 epochs as the validation loss no longer showed improvement for some time (Figure 2). We used the Adam optimizer with the mini-batch size of 256 and the learning rate of 0.01 halved every 20 epochs. The final validation root mean squared error (RMSE) was 10.4 and the training took 1.5 minutes on an Nvidia RTX 2080S GPU.

Evolution with the ANN. We then ran an evolution of grids using the trained ANN as a part of the fitness function.[‡] In the notation of Section IV the fitness function was $f(G) = \alpha \cdot l_G - s_G$ and we set $\alpha = 1000$. Evolutionary hyperparameters were the same as above. We ran evolution for 100 generations (Figure 1, bottom). We then removed duplicates from the final population and ran WOMBAT on the remaining 555 grids. WOMBAT scored 469 of them, finding full solutions for 14.9% of the grids. The average score was 128.19 (Table I).

VI. DISCUSSION

As per Figure 3 and Table I, the evolution with the ANN-predicted score produced grids which have substantially higher WOMBAT solution scores than either random grids or the grids evolved with the maximum word-slot-length as the fitness function. Additionally, ANN-guided evolution was the only method that generated any grids fully solvable by WOMBAT. We show such grid examples and their solutions in Figure 4.

This is not surprising since the ANN appears to be a better predictor of WOMBAT score than the maximum world-slot length (mWSL). Comparing both on the 100 random grids held out for testing, the Spearman's rank correlation coefficient between negative ANN score and WOMBAT score is -0.64 whereas for mWSL and WOMBAT score it is only -0.24. Thus evolutionary selection of grids with lower ANN score is more likely to increase WOMBAT score than by merely minimizing mWSL. Figure 5 shows the 100 individual data points as well as the least squares linear fit for both predictors.

TABLE I: Generating grids with different methods.

Method	Grids	Solved	Partial/full score (mean ± std)
random sampling	2092	0%	59.22 ± 13.14
mWSL evolution	1866	0%	95.70 ± 18.03
ANN evolution	938	14.9%	128.19 ± 21.17

VII. FUTURE WORK

A straightforward extension of this work is to increase the size of the training set for deep learning by generating more random grids and running them through WOMBAT. The resulting trained ANN is likely to be a better predictor of WOMBAT solution scores, guiding the evolution better. Furthermore, the specific ANN used is *ad hoc* in its topology leaving a number of questions for future investigation.

Independently, one can add the higher-quality grids evolved by ANN-guided evolution and their WOMBAT solution scores to the training set and re-train the ANN. Then the evolution can be repeated, hopefully producing yet higher-score grids. The iterative process can then be repeated. Future work can also investigate applicability of generative adversarial networks [8] to generation of valid high-score crossword grids.

It will also be of interest to re-evaluate the evolved grids (i.e., compute their solution scores) with WOMBAT with a higher time limit and a higher memory limit per puzzle instance. Since WOMBAT is under active development, future work will benefit from its stronger forthcoming versions. Finally, it would be of interest to evaluate evolved grids with competition-level human crossword solvers.

VIII. CONCLUSIONS

This work-in-progress paper presented an effective method for procedural generation of high-quality Romanian crossword puzzles. To do so we evolved crossword puzzle instances using a deep neural network as a part of the fitness function. Evaluating the evolved puzzles with a state-of-the-art AI crossword solver showed a substantial improvement over random grid sampling as well as over a basic evolution without the network.

[†]Occasionally WOMBAT does not produce even a partial solution due to exceeding the memory limit.

[‡]For each grid we ran ANN on it and on its transposition and then averaged the two predicted scores.

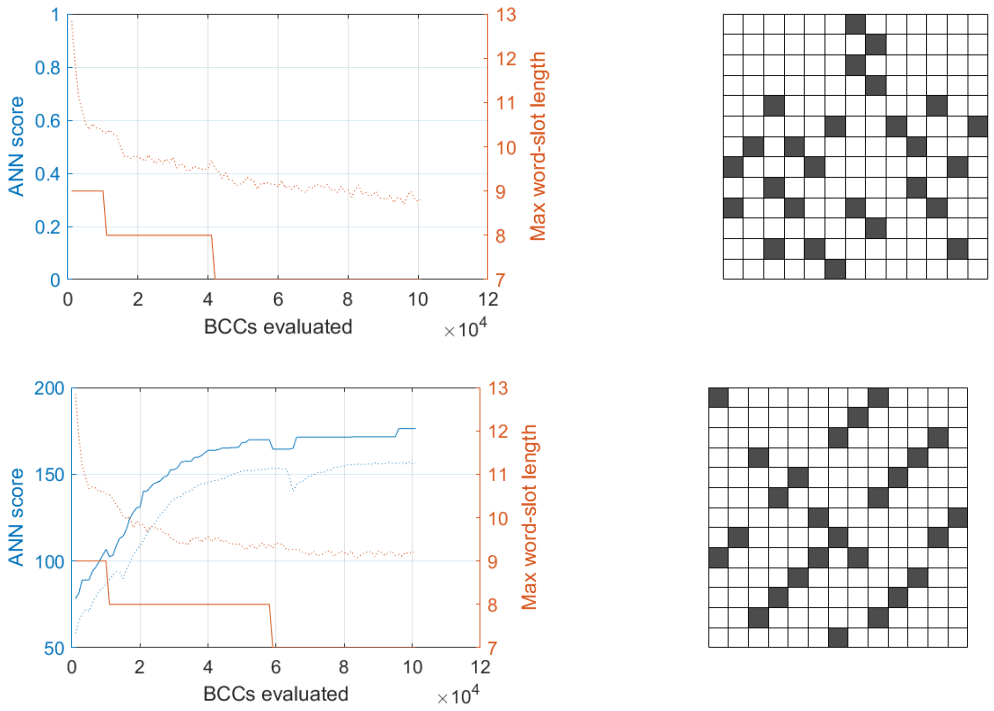

Fig. 1: **Top:** grid evolution with mWSL as the sole fitness function. The solid line shows mWSL for the best grid per generation. The dotted line shows population average. The fittest grid found is on the right. The x-axis is the number of grids whose fitness was computed. **Bottom:** Grid evolution with mWSL and ANN as the fitness function. The solid lines shows ANN and mWSL for the best grid per generation. The dotted lines show population averages. The fittest grid is shown on the right.

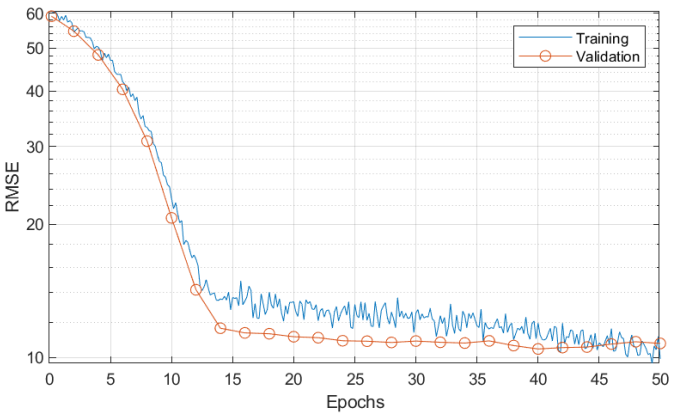

Fig. 2: Learning curves for ANN training.

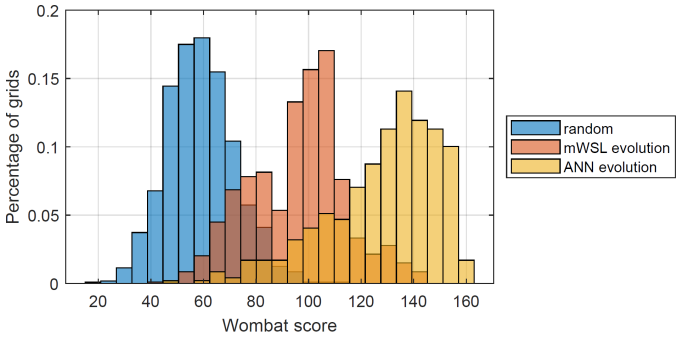

Fig. 3: Full/partial score distributions with different grid-generation methods.

Acknowledgment

We appreciate support from Compute Canada and the anonymous reviewers.

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1979.

[2] A. Botea and V. Bulitko, "Scaling up search with partial initial states in optimization crosswords," in *Proceedings of the Symposium on Combinatorial Search (SoCS)*, 2021.

[3] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley, "Procedural content generation: Goals, challenges and actionable steps," in *Dagstuhl Follow-Ups*, vol. 6, 2013.

[4] S. Risi and J. Togelius, "Neuroevolution in games: State of the art and open challenges," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 1, pp. 25–41, 2017.

[5] V. Bulitko, S. Carleton, D. Cormier, D. Sigurdson, and J. Simpson, "Towards positively surprising non-player characters in video games," in *Proceedings of the Experimental AI in Games (EXAG) Workshop at the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2017, pp. 34–40.

[6] V. Bulitko, K. Doucet, J. Simpson, and A. Flynn, "A-life for non-playable characters in video games," in *Event Proceedings for Late-breaking Abstracts at A-LIFE conference*, 2018.

[7] V. Bulitko, M. Walters, M. Cselinacz, and M. Brown, "Evolving NPC behaviours in A-life with player proxies," in *Proceedings of the Experimental AI in Games (EXAG) Workshop at the AAAI Conference*

■	P	I	T	T	A	R	D	■	L	U	E	S
S	I	M	I	A	N	D	■	B	E	N	D	A
C	E	N	T	R	E	■	D	U	C	A	■	R
H	R	■	R	E	C	L	A	S	A	■	G	T
M	I	C	U	■	S	I	V	E	■	V	A	R
I	T	I	■	V	I	Z	I	■	R	I	N	E
D	I	O	R	I	■	E	L	I	A	D	E	■
T	■	T	A	C	U	■	A	S	T	A	■	O
■	B	O	U	E	■	L	■	T	I	L	E	A
S	O	R	T	■	N	E	H	R	U	■	P	N
V	U	I	■	R	A	N	D	A	■	S	U	C
E	B	■	S	E	N	S	■	T	O	U	R	E
D	E	N	I	A	U	■	P	E	R	L	E	A

■	C	A	R	A	G	I	U	■	B	A	C	H
T	I	L	E	M	A	N	■	C	O	C	E	A
I	O	A	N	I	D	■	R	A	U	T	■	S
V	T	■	A	L	I	G	O	T	E	■	L	D
I	O	A	N	■	L	O	C	A	■	B	E	E
■	R	M	■	L	A	M	A	■	B	A	C	U
D	I	O	R	I	■	A	R	S	I	C	A	■
J	■	C	U	Z	A	■	D	U	C	A	■	S
U	T	■	L	E	N	S	■	T	I	L	E	A
V	U	I	A	■	T	A	C	U	■	B	O	R
A	D	N	■	I	E	D	E	■	S	A	N	T
R	O	S	E	T	■	E	L	I	A	S	■	R
A	R	A	F	A	T	■	E	L	I	A	D	E

■	B	E	A	T	R	I	X	■	B	A	C	H
P	E	R	C	I	U	N	■	M	A	T	E	I
A	R	E	I	C	■	V	O	I	C	U	■	O
L	L	■	R	A	D	E	S	C	U	■	L	T
L	I	G	A	■	I	L	I	U	■	S	E	T
A	E	R	■	S	E	I	S	■	T	A	C	U
D	R	A	G	U	■	T	O	C	I	L	A	■
Y	■	M	A	T	I	■	R	O	L	E	■	C
■	C	O	L	U	M	B	■	S	E	Y	N	I
G	A	N	E	■	N	A	F	T	A	■	O	O
O	R	T	■	V	■	L	E	E	■	T	I	R
M	O	■	M	A	S	S	U	■	D	U	C	A
A	L	L	A	I	S	■	D	O	C	S	A	N

Fig. 4: High-score ANN-evolved grids and their full solutions.

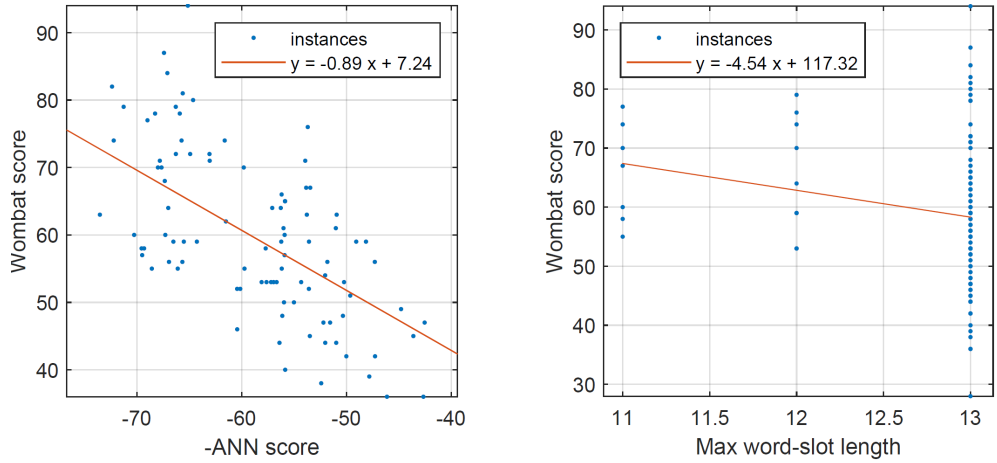

Fig. 5: WOMBAT scores and parts of the fitness function.

on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 2018.

[8] R. Rodriguez Torrado, A. Khalifa, M. Cerny Green, N. Justesen, S. Risi, and J. Togelius, "Bootstrapping conditional gans for video game level generation," in *Proceedings of the IEEE Conference on Games (CoG)*, 2020, pp. 41–48.

[9] S. Risi, J. Lehman, D. D'Ambrosio, R. Hall, and K. Stanley, "Combining search-based procedural content generation and social gaming in the petalz video game," in *Proceedings of Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2012.

[10] Xbox Wire Staff, "Forza Horizon 2: What's a drivatar, and why should I care?" *Xbox Wire*, 2014.

[11] Hello Games, "No Man's Sky Next," 2018.

[12] T. Soule, S. Heck, T. E. Haynes, N. Wood, and B. D. Robison, "Darwin's Demons: Does evolution improve the game?" in *Proceedings of European Conference on the Applications of Evolutionary Computation*, 2017, pp. 435 – 451.

[13] B. De Kegel and M. Haahr, "Procedural Puzzle Generation: A Survey," *IEEE Transactions on Games*, vol. 12, no. 1, pp. 21–40, 2020.

[14] D. Bonomo, A. P. Lauf, and R. Yampolskiy, "A Crossword Puzzle Generator Using Genetic Algorithms with Wisdom of Artificial Crowds," in *CGAMES*. IEEE Computer Society, 2015, pp. 44–49.

[15] M. L. Littman, G. A. Keim, and N. Shazeer, "A probabilistic approach to solving crossword puzzles," *Artificial Intelligence*, vol. 134, no. 1, pp. 23–55, 2002.

[16] M. L. Ginsberg, M. Frank, M. P. Halpin, and M. C. Torrance, "Search Lessons Learned from Crossword Puzzles," in *Proceedings of the National Conference on Artificial Intelligence*, 1990, pp. 210–215.

[17] A. Botea, "Crossword Grid Composition with A Hierarchical CSP Encoding," in *Proceeding of the 6th CP Workshop on Constraint Modelling and Reformulation, ModRef-07*, 2007.

[18] C. Lecoutre and O. Roussel, "Proceedings of the 2018 XCSP3 competition," *CoRR*, vol. abs/1901.01830, 2019.

[19] G. Audemard, C. Lecoutre, and M. Maamar, "Segmented tables: An efficient modeling tool for constraint reasoning," in *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, vol. 325, 2020, pp. 315–322.