Improving Human Players' T-Spin Skills in Tetris with Procedural Problem Generation

Taishi Oikawa[1], Chu-Hsuan Hsueh[1], and Kokolo Ikeda[1⋆]

School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan
{taishi_o,hsuehch,kokolo}@jaist.ac.jp

Abstract. Researchers in the field of computer games interest in creating not only strong game-playing programs, but also programs that can entertain or teach human players. One of the branches is procedural content generation, aiming to generate game contents such as maps, stories, and puzzles automatically. In this paper, automatically generated puzzles are used to assist human players in improving the playing skills for the game of Tetris, a famous and popular tile-matching game. More specifically, a powerful technique called T-spin is hard for beginners to learn. To assist beginners in mastering the technique, automatically generated two-step to T-spin problems are given for them to solve. Experiments show that the overall ability for beginners to complete T-spin during play is improved after trained by the given problems. The result demonstrates the possibility of using automatically generated problems to assist human players in improving their playing skills.

Keywords: Procedural Content Generation · Puzzle · Tetris · Training System · Entertainment.

1 Introduction

In recent years, artificial intelligence has made significant progress in computer board games as well as video games. Remarkably, computer programs can learn to surpass human levels without human knowledge of games. One example is the AlphaZero algorithm [17], which achieved superhuman levels of plays in three classical board games, chess, shogi, and Go. Another example for video games is that the programs based on deep reinforcement learning [8,10] obtained higher scores than professional human players in many of the Atari 2600 games.

Creating programs to entertain or teach human players is another popular research topic in the past decades. Hunicke and Chapman [4] proposed a probabilistic method to dynamically adjust the difficulty of a first-person shooter game. Ikeda and Viennot [6], and Sephton *et al.* [15] tried to create entertaining programs based on Monte-Carlo tree search (MCTS). Sephton *et al.* [15], Demediuk *et al.* [2], and Wu *et al.* [19] investigated strength and difficulty adjustment for MCTS-based programs. Ikeda *et al.* [5], Takahashi [18], and Oikawa

⋆ Corresponding author

and Ikeda [11] aimed to build training systems for the games of Go, Puyo-Puyo, and Tetris respectively.

This paper continues the work by Oikawa and Ikeda [11] to create a training system for Tetris, a famous tile-matching puzzle game. In Tetris, a technique called *T-spin* is powerful but hard to be learnt by beginners. As the first step to assist human players in mastering the technique, the previous work automatically generated *one-step to T-spin* problems. The generated problems were solved and rated by beginners in five-grade interestingness and difficulty. Generally, interestingness and difficulty were highly positively correlated.

In addition to generating problems, more emphasis in the paper is put on analyzing the effectiveness of training human players by the automatically generated problems. In the experiments, players are divided into three groups where one is trained by normal play and the other two by *two-step to T-spin problems* as well as normal play. After training, the win rates increase for all groups in a competition variant of Tetris. Especially, the two groups trained with T-spin problems can trigger T-spin more often. The results demonstrate the potential to build a training system for improving the playing skills of human players.

The rest of this paper is organized as follows. Section 2 briefly introduces the game of Tetris, including the technique of T-spin, and then reviews some work related to procedural problem generation. Sections 3 and 4 present the two-stage experiments where the latter is the key experiments in this paper. In the first stage, prediction models for interestingness and difficulty of two-step problems are built. The second stage incorporates the interestingness predictor to generate problems to train human players. The experiments aim to verify how T-spin problems assist human players in improving their playing skills. Finally, Section 5 makes concluding remarks and discusses future research directions.

2 Background

This section introduces the game of Tetris in Subsection 2.1 and related work on procedural problem generation in Subsection 2.2.

2.1 The Game of Tetris

Tetris is a kind of tile-matching puzzle video game presented in 1984 by Pajitnov[1]. The game has a widespread popularity and can be played on various platforms. There are many different variants of the game. This subsection briefly describes some basic elements of Tetris. The game is played on a rectangular field with twenty rows and ten columns. Seven kinds of *tetrominoes* are used, each of which is made up of four connected squares. The tetrominoes are named according to the similarity to Latin alphabet letters, I, J, L, O, T, S, and Z.

During play, tetrominoes fall down by a random sequence. One tetromino falls down at a time, with the next one(s) shown as a hint. The player can move

[1] https://en.wikipedia.org/wiki/Tetris

the given tetromino by one of the three directions left, right, and down, or rotate it by 90-degree. The tetromino stops falling when at least one square meets the bottom line or other occupied squares below, as shown by the bright gray squares in Figures 1a and 1b in smaller fields. When all squares in a row are occupied, the row is cleared and the player obtains some bonus scores. An example of clearing a single row is depicted in Figure 1c where the second row from the bottom is cleared. It is possible to clear at most four rows at a time, and higher scores are obtained when more rows are cleared. The squares above the cleared row(s) then fall down with the shape remained, as shown in Figure 1d. A game ends when the occupied squares reach the top of the playing field and no tetromino can be further inserted. The goal of Tetris is to clear rows to obtain high scores. In some competition variants of Tetris, clearing rows on the player's own field generates *garbage* on the opponent field from the bottom, which may push the opponent closer to the end of games.

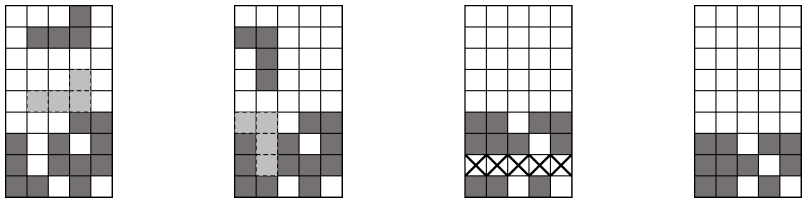
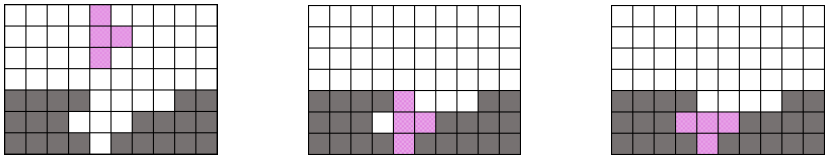


Fig. 1: An example of actions and row clearing in Tetris: (a) the current state, (b) manipulating the given L-tetromino, (c) clearing a row, and (d) the state after the row cleared.

A technique called *T-spin* is to insert a T-tetromino into a tight space by rotation immediately after the T-tetromino stops falling. An example of T-spin is shown in Figure 2. For the state in Figure 2a, if the T-tetromino falls down directly, a square in the second row from the bottom remains empty, as shown in Figure 2b. At the moment, the T-tetromino can be rotated clockwise by 90-degree, and then the state becomes the one in Figure 2c. In this example, two rows are cleared after T-spin, which is called *T-spin Double*. It is possible to clear at most three rows with T-spin, which is called *T-spin Triple*. Clearing rows with T-spin obtains extra bonuses; moreover, in competition variants of Tetris, more garbage is generated on the opponent field.

2.2 Related Work on Procedural Problem Generation

Procedural Content Generation (PCG) [16] is a research branch aiming to automatically create *game contents* by computer programs with limited or indirect human input. The term game content covers a large variety of components and



(a) (b) (c)

Fig. 2: An example of T-spin Double: (a) the current state, (b) falling down the T-tetromino, and (c) rotating the stopped T-tetromino to finish T-spin Double.

may refer to maps, levels, stories, and puzzle problems. This paper focuses on generating puzzle problems [1] and reviews some related work as follows.

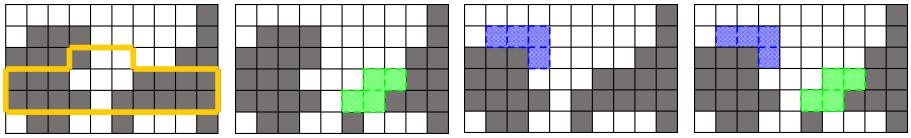
Hirose *et al.* [3] proposed to compose tsume-shogi problems (mating problems in shogi) by *reverse method*, which started from board states with checkmate and then searched reversely for n moves. Two of the generated problems were introduced in a tsume-shogi magazine and received favourable feedback. For chess mating problems, Schlosser [14] automatically composed based on endgame tables built up by retrograde analysis. Iqbal [7] improved the efficiency to generate mate-in-3 problems with incorporating probabilities of piece locations derived from different databases.

Mantere and Koljonen [9] applied genetic algorithm (GA) to not only solve but also generate and rate Sudoku puzzles. For another puzzle game called Shinro, Oranchak [12] showed that GA was effective to construct puzzles. Ortiz-García *et al.* [13] proposed to automatically generate picture-logic puzzles (also known as nonograms) from RGB color images. Takahashi [18] compared random method and reverse method on generating problems for Puyo-Puyo, where the later was shown to be more efficient.

Oikawa and Ikeda [11] proposed to generate n-step to T-spin problems by reverse method. The approach is briefly summarized as follows. For the simplicity of discussion, T-spin in the rest of this paper refers to T-spin Double if not specified. First, basic patterns with a complete T-shape was created, as the part highlighted in Figure 3a. To enrich the diversity of the generated T-spin patterns, noises were introduced. An example of the resulted T-spin pattern is depicted in Figure 3a. Reverse method then generated n-step to T-spin problems by removing n tetrominoes from the playing field one at a time. Figures 3b and 3c show two examples of one-step to T-spin problems with removing the highlighted S-tetromino and J-tetromino respectively. A two-step to T-spin problem with S- and J-tetrominoes is shown in Figure 3d. In their experiments, one-step to T-spin problems were generated and analyzed. They showed that interestingness and difficulty of the problems had a highly positive correlation for beginners.

3 Experiments on Interestingness and Difficulty

As a preliminary study, a total of sixteen players (males between 22 and 27 years) participated in the experiments. All of them were interested in playing



(a) (b) (c) (d)

Fig. 3: An example of the reverse method for T-spin problems: (a) a T-spin pattern, (b) a one-step problem with S-tetromino, (c) a one-step problem with J-tetromino, and (d) a two-step problem with S- and J-tetrominoes.

games, among which one was an expert of Tetris and the rest were beginners. To verify how two-step to T-spin problems assisted in improving the playing skills of beginners, the experiments were designed with two parts. In the first part, as described in more details in this section, predictors on interestingness and difficulty of two-step problems were built by data collected from the beginners. In the second part, beginners were divided into three groups, where one was trained without T-spin problems, one with randomly generated problems, and the other with interesting problems. The details are included in Section 4.

In this section, the settings of experiments on interestingness and difficulty, the features extracted for two-step to T-spin problems, and the results are presented in Subsections 3.1, 3.2, and 3.3 respectively.

3.1 Experiment Settings

First, the T-spin technique and its importance to Tetris were explained to the beginners. They then solved 50 two-step problems with the tool shown in Figure 4a. All players solved the same set of problems but in random order.

(a) (b)

Fig. 4: (a) Tool for the experiments and (b) an example for the new features in two-step problems.

After pressing the "Start" button, 50 problems were given one at a time. At the beginning of a problem, the given tetromino was put in the playing field, with the next one shown on the right-hand side, as illustrated in Figure 4a. Players could drag, rotate, and drop the given tetromino. After pressing the "Confirm" button, the location of the first tetromino was determined and could not be changed. The second tetromino was then put into the playing field. After determining the location of the second tetromino, players rated the interestingness and the difficulty in five-grade evaluation. Scores 1 to 5 represented most uninteresting/easiest to most interesting/difficult. Solutions could also be obtained by pressing the "Show Answer" button. Players then pressed the "Next" button to solve the next problem until all problems were finished. The reason to enable players to obtain the solutions was to prevent the players from giving up due to frustration and rating the problems arbitrarily.

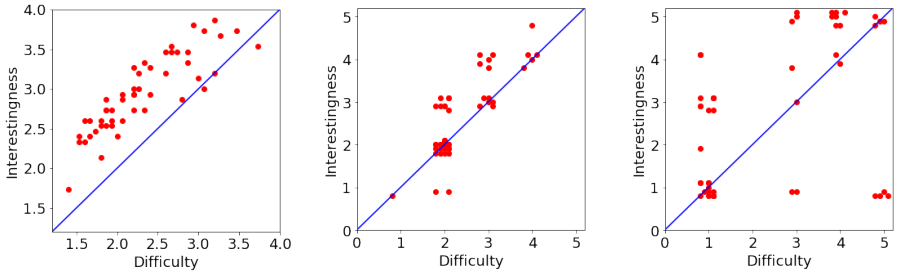
3.2 Features of Two-Step to T-Spin Problems

Oikawa and Ikeda [11] have proposed some features for one-step problems. Two-step problems are supposed to be more difficult than one-step problems. Some problems may require the cooperation between the two given tetrominoes, as shown in Figure 4b. In order to better predict the interestingness and difficulty, four additional features are designed in this paper: (1) the number of edges connected between the two given tetrominoes, (2) the number of squares of the second tetromino that are located above the first one, (3) the number of squares of the two given tetrominoes that are located in the two rows to be cleared by T-spin, and (4) the number of squares of the two given tetrominoes that are located below the T-shape. The values of the four features for the example in Figure 4b are 1, 3, 4, and 2 respectively.

3.3 Results of Predictors

For each problem, the interestingness and the difficulty were averaged from the fifteen beginners. The results are plotted in Figure 5a. The correlation coefficient between interestingness and difficulty was 0.86, which showed a highly positive correlation. When looking into the ratings player by player, different preference can be found. This paper comments on the results of two players, as shown in Figures 5b and 5c with the data scattered for better display. For player A, no problems were rated as difficulty 5, and difficult problems tended to be interesting. Player B tended to rate difficult problems as either very interesting or very uninteresting. The player also rated easy problems as different interestingness.

It is expected that problems too difficult become uninteresting, and Player B did rate some difficult problems as very uninteresting. However, the overall interestingness and difficulty for two-step problems still had a high correlation. One possible explanation is that two-step problems were not too difficult for most of the beginners. The decrease of interestingness may occur for problems with more steps which are supposed to be even more difficult.



(a) (b) (c)

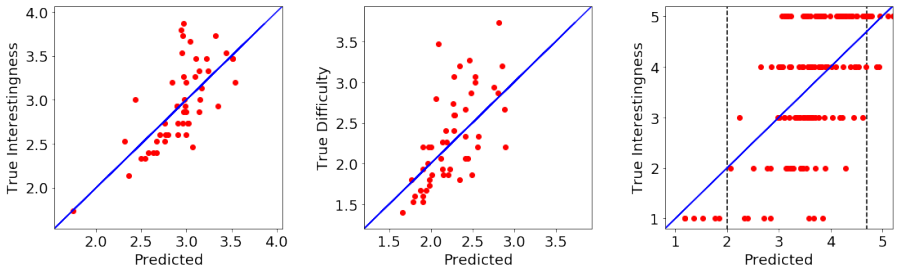
Fig. 5: Relations between interestingness and difficulty for (a) fifteen beginners averaged, (b) player A, and (c) player B.

To automatically generate interesting problems to train human players, a model to rate the interestingness of problems is required. In the experiments, supervised learning was applied to learn the interestingness. Ratings averaged from fifteen beginners for a total of 50 problems were used as the training data, with the features described in Subsection 3.2. The predictor was built based on the LightGBM framework[2] with 10-fold cross-validation. The results are plotted in Figure 6a, and the symmetric mean absolute percentage error (SMAPE) and mean absolute error (MAE) were 8.74% and 0.27 respectively. The model was able to predict the interestingness of two-step problems well. The results of the difficulty predictor are shown in Figure 6b, with a little higher SMAPE of 14.50% and MAE of 0.34. The model performed better for easier problems.

Figure 7a shows a problem which was predicted to be interesting by the interestingness predictor. Players may feel that the center part is a little bit empty and doubt whether it is possible to complete T-spin. The two L-tetrominoes, no matter the order, both locate beside the T-spin pattern. Once the solution is found, players may feel more satisfied and interesting.

One additional experiment was to ask the expert to solve and rate 147 two-step to T-spin problems. The results in Figure 6c shows that it was also possible to create interestingness predictors for individuals even experts, though the SMAPE of 26.17% and MAE of 0.84 were higher. One possible reason is that the granularity of the ratings for a single player is larger than the averaged values. Still, the model can be used to distinguish problems with interestingness 1 by predicted values lower than 2 and interestingness 4 and 5 by predicted values higher than 4.7. A problem considered interesting by the predictor is shown in Figure 7b. At the first glance, the expert thought that T-spin could be completed. However, the way that the expert considered resulted in a T-spin Single, which was thought as a trap and made the expert feel interesting. The experiment demonstrated the potential to personalize training systems.

[2] https://lightgbm.readthedocs.io/en/latest/



(a) (b) (c)

Fig. 6: Results of predictors on (a) interestingness and (b) difficulty for beginners, and (c) interestingness for an expert.

(a) (b)

Fig. 7: Interesting problems for predictors on (a) beginners and (b) an expert.

4 Experiments on Improving Human Players' Skills

This section presents the key experiments in this paper to demonstrate how two-step to T-spin problems assist human players in improving their T-spin skills in Tetris. Subsection 4.1 describes the experiment settings. The results and some discussions are then included in Subsection 4.2.

4.1 Experiment Settings

The flow chart of the experiments is shown in Figure 8. At the beginning, the strength of the players was measured by two approaches. First, they played best-of-three matches against three built-in CPU agents (Arle, Ess, and Zed) in Puyo-Puyo Tetris on the Steam platform[3]. Puyo-Puyo Tetris is a competition variant of Tetris, and the three CPU agents have different strength. Each player played at least six and at most nine games. The numbers of wins and losses, the time, the number of T-spin, and the whole progress of the games were recorded. The average time to finish a best-of-three match was 448 seconds with the standard deviation of 231 seconds. The actual time varied a lot from players, with the shortest and longest time of 161 and 1,354 seconds respectively.

In addition, the players solved fifteen two-step to T-spin problems selected by an expert. Each of the three levels, easy, middle, and difficult from the expert's

[3] https://store.steampowered.com/app/546050/Puyo_PuyoTetris

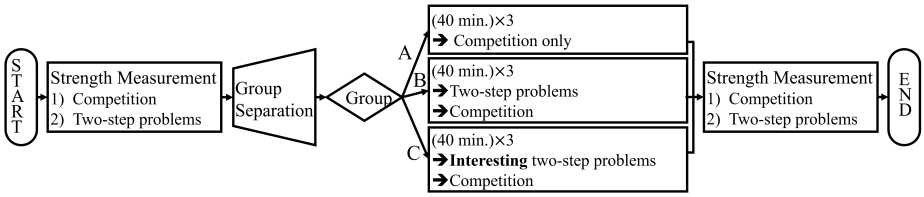


Fig. 8: Flow chart of the experiments on training human players.

view, contained five problems. The tool for solving problems was similar to the one shown in Figure 4a, except that the time limit for each problem was set to 30 seconds and the players could not obtain the solutions.

To investigate whether T-spin problems assisted players in improving playing skills, three different kinds of training menus were designed, one without T-spin problems while the other two with T-spin problems. Fifteen players were evenly divided into three groups such that the strength of the groups was similar. By strength of a group, it was the sum of the number of wins against the CPU agents and the number of correctly solved problems. The numbers are listed in Table 1, where "#W", "#G", "F_{TS}", and "#Sol" represent the number of wins, the number of played games, the frequency of T-spin per minute during play, and the number of correctly solved problems respectively. The three menus were given to the groups randomly. One set of training lasted for 40 minutes. All groups underwent the given sets of training three times. Namely, all the players were trained for 120 minutes.

Table 1: Results for groups A, B, and C before and after training.

	A (Competition Only)				B (Two-Step Problems)				C (Interesting Problems)			
	#W	#G	F_{TS}	#Sol	#W	#G	F_{TS}	#Sol	#W	#G	F_{TS}	#Sol
Before	7	35	0.21	59	4	33	0.24	62	5	32	0.18	61
After	13	33	0.13	60	11	36	0.75	67	8	33	0.79	61

The players in group A played against all three CPU agents Arle, Ess, and Zed during the whole 40 minutes in one set of training. For groups B and C, players were trained by 75 two-step to T-spin problems first. A similar tool to Figure 4a was used except that the time limit for each problem was set to 20 seconds and the solutions were always displayed after the players solved the problems. Group B was provided with problems randomly generated, while group C with interesting problems rated by predictor for beginners. Training by T-spin problems took about 20 minutes. In the rest of the time in 40 minutes, the players in both groups played against CPU agents as group A. After three sets of training were finished, the strength of the players was measured again, with the order of the fifteen problems changed.

4.2 Results of Skill Improvement

The results for the three groups after training are listed in Table 1. The win rates of all the groups increased after training, which were 20.0±13.3% to 39.4±16.7%, 12.1±11.1% to 30.6±15.0%, and 15.6±12.6% to 24.2±14.6% respectively. Although the result of group C was not statistically significant, the experiments still showed that the strength of the players with regard to the competition variant could be improved after training. The correctness rates for T-spin problems of 78.7±9.3% to 80.0±9.1%, 82.7±8.6% to 89.3±7.0%, and 81.3±8.8% to 81.3±8.8% were not improved significantly, even for groups B and C.

However, when focusing on the frequency of T-spin, the growth rates for groups B and C were 212% and 334% respectively, while group A was -39%. The frequencies of T-spin per minute for the fifteen players are plotted in Figure 9. Generally, players in groups B and C improved their T-spin skills. Especially, for those who completed less T-spin before training, the growth was even clearer. Interestingly, a player in group A tended not to perform T-spin after training.

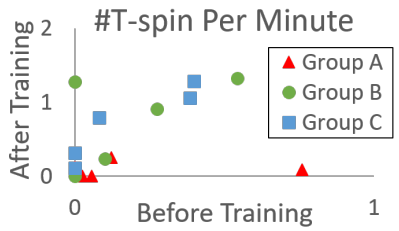


Fig. 9: Frequency of T-spin per minute before and after training.

To investigate the reason why players with higher T-spin skills did not win more games (groups B and C vs. group A), the recorded progress of the games was reviewed by an expert. It was found that the speeds to arrange the tetrominoes were reduced while the players attempted to create T-spin patterns. Sometimes, they also made mistakes on the arrangement of tetrominoes. As a result, the merits obtained from T-spin were almost eliminated by the speed reduction and the mistakes. If the players were trained for a longer time to be familiar with T-spin patterns, their strength may improve more significantly. Overall, the experiments suggested that the strength of players could be improved after training. Especially, players trained with T-spin problems improved the strength with aware of the technique.

5 Conclusions and Future Work

In this paper, two-step to T-spin problems are generated automatically to train beginners. First, models trained by ratings collected from beginners can predict

the interestingness of two-step problems well. In addition, it is also possible to train models to predict interestingness of problems for individual players. This demonstrates the potential to personalize training systems of games.

Moreover, experiments on training players with three different menus are conducted. Overall, the strength of the players is improved. Especially, players trained with T-spin problems indeed complete T-spin more frequently, though they do not improve win rates more than the players trained without T-spin problems. It is supposed that the difference will be bigger if the training time became longer for the players to master their T-spin skills.

The followings discuss several promising research directions. With the success in predicting the interestingness of one-step and two-step to T-spin problems. the next step to the training system is to try three-step or four-step problems, which are even closer to real game-play. It is also interesting to find the point that difficult problems become uninteresting for beginners. In addition, some more techniques such as clearing four rows and T-spin Triple are also important to Tetris. Similar approaches can be applied to build training systems for these techniques. It is also possible to mix different techniques in a training system.

Acknowledgments. This research is financially supported by Japan Society for the Promotion of Science (JSPS) under contract numbers 18H03347 and 17K00506.

References

1. De Kegel, B., Haahr, M.: Procedural puzzle generation: A survey. IEEE Transactions on Games pp. 1–1 (2019). https://doi.org/10.1109/TG.2019.2917792
2. Demediuk, S., Tamassia, M., Raffe, W.L., Zambetta, F., Li, X., Mueller, F.: Monte Carlo tree search based algorithms for dynamic difficulty adjustment. In: 2017 IEEE Conference on Computational Intelligence and Games (CIG 2017). pp. 53–59. IEEE (2017). https://doi.org/10.1109/CIG.2017.8080415
3. Hirose, M., Ito, T., Matsubara, H.: Automatic composition of tsume-shogi by reverse method. Journal of Japanese Society for Artificial Intelligence **13**(3), 452–460 (1998)
4. Hunicke, R., Chapman, V.: AI for dynamic difficulty adjustment in games. In: AAAI-04 workshop on Challenges in Game Artificial Intelligence. pp. 91–96. AAAI Press (2004)
5. Ikeda, K., Shishido, T., Viennot, S.: Machine-learning of shape names for the game of Go. In: Plaat, A., van den Herik, H.J., Kosters, W. (eds.) The 14th International Conference on Advances in Computer Games (ACG 2015). pp. 247–259. Springer International Publishing (2015). https://doi.org/10.1007/978-3-319-27992-3_22
6. Ikeda, K., Viennot, S.: Production of various strategies and position control for Monte-Carlo Go - entertaining human players. In: 2013 IEEE Conference on Computational Inteligence in Games (CIG 2013). pp. 145–152. IEEE (2013). https://doi.org/10.1109/CIG.2013.6633625
7. Iqbal, A.: Increasing efficiency and quality in the automatic composition of three-move mate problems. In: Anacleto, J.C., Fels, S., Graham, N., Kapralos, B., Saif

El-Nasr, M., Stanley, K. (eds.) The 10th International Conference on Entertainment Computing (ICEC 2011). pp. 186–197. Springer Berlin Heidelberg (2011). https://doi.org/10.1007/978-3-642-24500-8_20
8. Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., Dabney, W.: Recurrent experience replay in distributed reinforcement learning. In: The Seventh International Conference on Learning Representations (ICLR 2019) (2019)
9. Mantere, T., Koljonen, J.: Solving, rating and generating Sudoku puzzles with GA. In: 2007 IEEE Congress on Evolutionary Computation (CEC 2007). pp. 1382–1389. IEEE (2007). https://doi.org/10.1109/CEC.2007.4424632
10. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
11. Oikawa, T., Ikeda, K.: Procedural problem generation of Tetris for improving T-spin skill. In: The 23rd Game Programming Workshop (GPW-18). pp. 175–182 (2018)
12. Oranchak, D.: Evolutionary algorithm for generation of entertaining Shinro logic puzzles. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekrt, A., Esparcia-Alcazar, A.I., Goh, C.K., Merelo, J.J., Neri, F., Preu, M., Togelius, J., Yannakakis, G.N. (eds.) European Conference on the Applications of Evolutionary Computation (EvoApplications 2010). pp. 181–190. Springer Berlin Heidelberg (2010). https://doi.org/10.1007/978-3-642-12239-2_19
13. Ortiz-García, E.G., Salcedo-Sanz, S., Leiva-Murillo, J.M., Pèrez-Bellido, A.M., Portilla-Figueras, J.A.: Automated generation and visualization of picture-logic puzzles. Computers & Graphics **31**(5), 750–760 (2007). https://doi.org/10.1016/j.cag.2007.08.006
14. Schlosser, M.: Computers and chess problem composition. ICCA Journal **11**(4), 151–155 (1988). https://doi.org/10.3233/ICG-1988-11404
15. Sephton, N., Cowling, P.I., Slaven, N.H.: An experimental study of action selection mechanisms to create an entertaining opponent. In: 2015 IEEE Conference on Computational Intelligence and Games (CIG 2015). pp. 122–129. IEEE (2015). https://doi.org/10.1109/CIG.2015.7317939
16. Shaker, N., Togelius, J., Nelson, M.J.: Procedural Content Generation in Games: A Textbook and an Overview of Current Research. Springer (2016)
17. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T.P., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science **352**(6419), 1140–1144 (2018). https://doi.org/10.1126/science.aar6404
18. Takahashi, R.: Mating problem generation of Puyo-Puyo for training. Master thesis, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan (2018)
19. Wu, I.C., Wu, T.R., Liu, A.J., Guei, H., Wei, T.h.: On strength adjustment for MCTS-based programs. In: The 33rd AAAI Conference on Artificial Intelligence (AAAI-19). AAAI Press (2019)