Proceedings, The Thirteenth AAAI Conference on
Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)

Level Difficulty and Player Skill Prediction in Human Computation Games

Anurag Sarkar, Seth Cooper
College of Computer and Information Science
Northeastern University
Boston, Massachusetts, USA
sarkar.an@husky.neu.edu, scooper@ccs.neu.edu

Abstract

Human computation games (HCGs) often suffer from low player retention. This may be due to the constraints placed on level and game design from the real-world application of the game. Previous work has suggested using player rating systems (such as Elo, Glicko-2, or TrueSkill) as a basis for matchmaking between HCG levels and players, as a means to improve difficulty balancing and thus player retention. Such rating systems typically start incoming entities with a default rating. However, when applied to HCGs, incoming entities may have useful information associated with them, such as player behavior during tutorials and properties of the tasks underlying the levels. In this work, we examined using features derived from player behavior and level properties to predict their eventual Glicko-2 ratings in the HCG *Paradox*. We found that using regression produced rating estimates closer to the actual ratings than default or baseline average ratings. The use of rating systems allows a unified approach to predicting both player skill and level difficulty.

Introduction

Human computation games (HCGs) often suffer from low player retention, with most players leaving soon after starting, and the bulk of work being accomplished by a small fraction of players (Sturn et al. 2015). This contribution pattern holds true in other human computation systems, such as online citizen science projects (Sauermann and Franzoni 2015). Improving retention of members of the "long tail" of participants may help in increasing the effectiveness of human computation systems in general. This, in turn, may help advance the range of problems to which HCGs can be applied, which have included image labeling (von Ahn and Dabbish 2004), graph-theoretic problems (Cusack et al. 2010), neuron reconstruction (Kim et al. 2014), gathering common-sense facts (Siu and Riedl 2016), and protein folding (Cooper et al. 2010).

Due to the constraints of solving pre-existing problems, HCG designers do not have complete freedom to modify levels. Previous work (Cooper, Deterding, and Tsapakos 2016) has suggested that these constraints may contribute to low retention, and has suggested using existing player rating systems as a basis for matchmaking between HCG levels and players, as a means to improve difficulty balancing and thereby player retention. Rating systems—such as Glicko-2 (Glickman 2001)—are often used for player-versus-player matchmaking, though they have been used in other applications involving pairwise comparison of entities, including player-versus-level matchmaking. In such systems, each entity has a rating, which is updated based on the result of some comparison (for example, a player attempting a level, and either winning or losing).

Rating systems typically start incoming entities (i.e. players or levels) with a default rating. They are essentially treated all the same—a "blank slate" about which there is no information. However, in the case of HCGs, there may be some information about incoming entities that can be used to inform their initial rating. For players, this can include their behavior in the game up to the point where a rating is needed—for example, their performance in any tutorials or onboarding before they are to be matched with a level derived from a real task. For levels, this can include various properties of the underlying task—for example, size, estimates of complexity, and so on.

In this work, we gathered gameplay data in the HCG *Paradox* from players recruited using Mechanical Turk. We used this data to assign ratings to the players and levels using the Glicko-2 system. For use in a rating system, we treated each instance of a player successfully completing a level as a win for the player, and a player failing to complete a level as a win for the level. Thus, the most difficult levels (i.e. the levels that players most often fail to complete) are those that "win" the most, and therefore end up with the highest ratings, similar to a player-vs-player setting where the players with the highest ratings are those that win the most matches against other players. Hence, using rating systems offers a reasonable way of assessing both level difficulty and player skill, giving us a suitable ground truth on which to make our predictions.

We then used linear and Gaussian process regression to predict the ratings of players—based on their behavior in the game's tutorial levels—and levels—based on properties of the underlying problem defining the level. We found that rating predictions based on regression were closer to the actual ratings than both the default Glicko-2 rating and a baseline method using average ratings. Predictions generally performed better for level ratings than for player ratings.

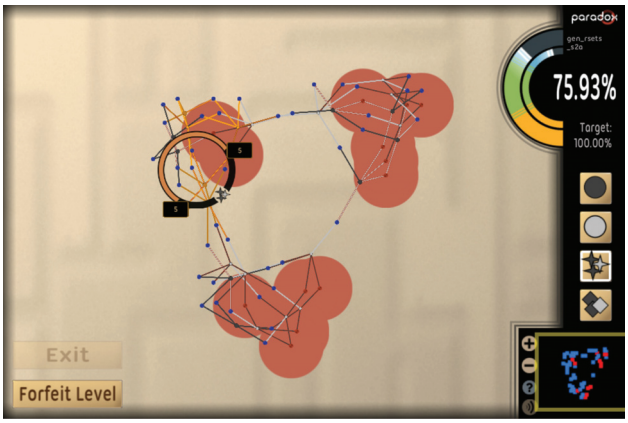

Figure 1: Screenshot from the HCG *Paradox*.

Related Work

Rating Systems

Player rating systems are used to match up players possessing comparable levels of skill. Several such rating systems exist, with Elo (Elo 1978), Glicko (Glickman 1999), Glicko-2 (Glickman 2001), and TrueSkill (Herbrich, Minka, and Graepel 2007) being some notable examples. While most commonly used for player-versus-player skill rating and matchmaking, these systems are increasingly being applied to other situations to handle the results of pairwise (Hacker and von Ahn 2009) or setwise (Sarkar et al. 2016) comparisons.

Such rating systems are usually employed in settings (e.g. chess, online gaming) where no useful information is known about new, unrated players who are thus assigned a uniform default rating. However, in the domain of HCGs, we may have access to information about players (or levels) that could help the rating system assign a starting rating which may be more indicative of the player's actual skill (or the level's actual difficulty) than the default rating. It is this research question that we primarily explore in this work.

Level Difficulty Prediction

Several approaches to predicting level difficulty have been developed. Closely related to our work is that of van Kreveld et al. (2015), who used a linear function to predict the difficulty (on a subjective scale of 1-10) of levels in various puzzle games. Their predictions relied on, in addition to *initial* (static) features of a level, both *solution* features and *dynamic* (gameplay) features. Other work has examined, for example, estimating the difficulty of Sokoban levels using approaches such as the time taken to solve them, either automatically (Ashlock and Schonfeld 2010) or by players (Jarušek and Pelánek 2010). In our work on the case of HCGs, features based on level solutions or gameplay may not be present, as the solution to an HCG level may be unknown, and we wish to make predictions before any players play a level. Additionally, in our work, we predict player skill using the same approach as level difficulty.

Levels in *Paradox* are derived from instances of the maximum satisfiability (MAX-SAT) problem. Thus, predicting the level difficulty is related to predicting the difficulty of the underlying MAX-SAT problem. Existing work has examined understanding what features of SAT problems make them hard. These include features such as the average number of variables per clause (Mitchell, Selman, and Levesque 1992) and tree-related subgraph features (Mateescu 2011). Several of the level features used in our work are informed by these.

Dynamic Difficulty

Although the immediate goal of our work in this paper is predicting player and level ratings, this will be applied to matchmaking between players and levels, with the goal of assigning levels of appropriate difficulty to players based on their skill. This relates to dynamic difficulty adjustment (Hunicke 2005), which has received considerable attention in the game research community. Inspired by the *flow* theory of Csikszentmihalyi (1990), dynamic difficulty adjustment can attempt to affect player engagement by personalizing the difficulty of a game for each player's skill. Much work has examined the relationship between skill, difficulty, and engagement (Engeser and Rheinberg 2008; Alexander, Sear, and Oikonomou 2013; Denisova and Cairns 2015; Lomas et al. 2013).

Data Collection

Paradox (Dean et al. 2015) is a 2D puzzle HCG originally designed for the purpose of crowdsourced formal verification of software. Each level in the game represents a MAX-SAT problem, which the players attempt to solve by making use of both manual and automated tools (represented as "brushes" in-game) to assign values to different variables. Players are assigned a score based on the percentage of clauses they are able to satisfy and can "complete" a level by reaching a pre-determined target percentage.

The game starts with a set of 9 hand-crafted tutorial levels, intended to introduce the game, which include instructions and hints on gameplay. After completing the tutorial, players then proceed to the challenge levels derived from MAX-SAT problems. In this work, we used levels generated from a number of MAX-SAT problems encoded in the DIMACS file format. We decided upon a set of 50 levels, 38 of which were derived from the SATLIB Benchmark Problems[1] while the remaining 12 were generated using randomized algorithms.

For player recruitment, we used a Human Intelligence Task (HIT) posted on Amazon Mechanical Turk (MTurk) that paid $1.50 upon completion. Through this HIT, we recruited 150 players. Players were required to complete all tutorial levels, after which they advanced to the challenge section where they were served the 50 aforementioned levels in random order. Players were not served the same level more than once until they had seen all of the 50 levels. They did not have to complete these challenge levels, and had the option of skipping (moving to the next level without making

[1] http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html

	Default	*Average*	*LR*	*GPR*
Players	211.1	198.0	195.2	197.3
Levels	403.8	412.0	358.9	319.2

Table 1: Root Mean Square Error (RMSE) in rating predictions for players and levels.

a move) or forfeiting (moving to the next level after making at least 1 move) them. Upon skipping or forfeiting at least 5 levels, players could finish the HIT.

The match data from our HIT was then played back into the Glicko-2 rating system, using the `pyglicko2` Python module (Kirkman 2010). Both players and levels were started out with default Glicko-2 parameters (rating of 1500, rating deviation of 350 and volatility of 0.06). We treated each instance of a player seeing a level as a match in order to generate the ratings used in our analysis. This resulted in three possible outcomes: a player completing a level (i.e. reaching the target percentage) counted as a win for the player, a player forfeiting a level counted as a loss for the player, and a player skipping a level was ignored for the purpose of rating generation.

From the initial pool of 150 players, we filtered out from further analysis those who had either played fewer than 5 levels, hadn't completed the tutorial, or had only skipped through the levels. After doing so, we arrived at a player count of 99. We did not filter out any of the 50 levels, as each was involved in at least 15 matches.

Analysis

To predict ratings for players and levels, we derived several features to be used in regression. The features used for players were based on their behavior during the game's tutorial levels. This is data that would be available *before* players started the game's challenge levels and thus before they need to be assigned a rating. The following three features were aggregated across all tutorial levels:

- Total time taken (in seconds) to complete all tutorial levels.
- Total number of moves used to complete all tutorial levels.
- The amount by which the player's total score for the tutorials exceeded the total target score of the tutorials.

The features used for levels were based on static properties of the levels themselves. These included properties of the underlying MAX-SAT problem:

- Average number of variables per clause.
- Average number of clauses per variable.
- 1 if all clauses are satisfied when all variables are set to true, otherwise 0.
- 1 if all clauses are satisfied when all variables are set to false, otherwise 0.
- Percentage of clauses satisfied when all variables are set to true.
- Percentage of clauses satisfied when all variables are set to false.

	Average	*LR*	*GPR*
Players	66%	66%	65%
Levels	22%	66%	68%

Table 2: Percentage of predictions closer to the actual rating than baseline *default* predictions.

	LR	*GPR*
Players	55%	58%
Levels	68%	68%

Table 3: Percentage of predictions closer to the actual rating than baseline *average* predictions.

- Absolute value of the difference of the previous two features.

as well as properties of the factor graph (Sinz 2007) representation of the underlying MAX-SAT problem:

- Number of nodes.
- Number of variable nodes.
- Number of clause nodes.
- Number of edges.
- Percentage of edges in minimum spanning tree.
- Size of the periphery. (Periphery of a graph is the set of nodes having eccentricity equal to the graph's maximum eccentricity. The eccentricity of a node is the maximum distance from that node to all other nodes in the graph.)
- Average shortest path length—computed using the equation

$$\sum_{s,t \in V} \frac{d(s,t)}{n(n-1)}$$

where V is the set of nodes in the graph, $d(s,t)$ is the shortest path between nodes s and t, and n is the number of nodes in the graph.

We thus ended up with 14 features for the levels.

For our analysis, we used the player and level features described above, along with the Glicko-2 ratings obtained from the HIT, to train and test different approaches for predicting ratings. To fit regression models on the feature data for the purpose of ratings predictions, we used the `scikit-learn` Python module (Pedregosa et al. 2011). We used the following prediction methods:

- *Default*: Always predicting the default rating (of 1500). This was a baseline for comparing to using default values.
- *Average*: Predicting the average of all training ratings. This was a simple baseline prediction.
- *LR*: Using linear regression for predictions. `scikit-learn`'s `LinearRegression` was used.
- *GPR*: Using Gaussian process regression for predictions. `scikit-learn`'s `GaussianProcessRegressor` was used, with a kernel consisting of RBF, constant, and noise terms. Each feature was offset and scaled so that the minimum value in the training set was 0 and the maximum value was 1. Ratings were scaled by dividing by 2000.

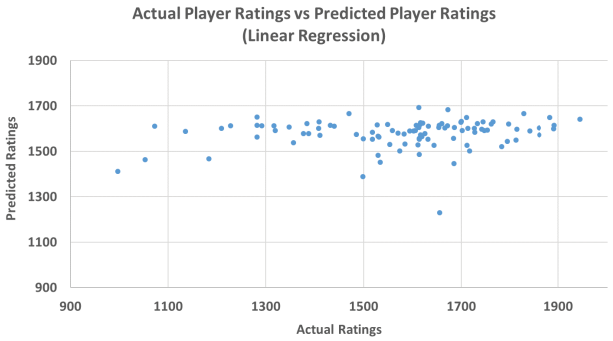

Figure 2: Scatter plot of actual and linear regression predicted ratings for players.

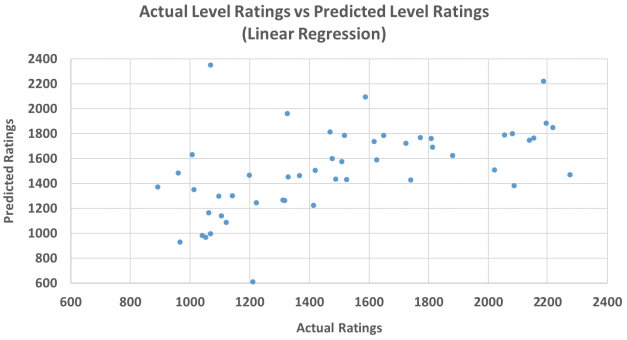

Figure 3: Scatter plot of actual and linear regression predicted ratings for levels.

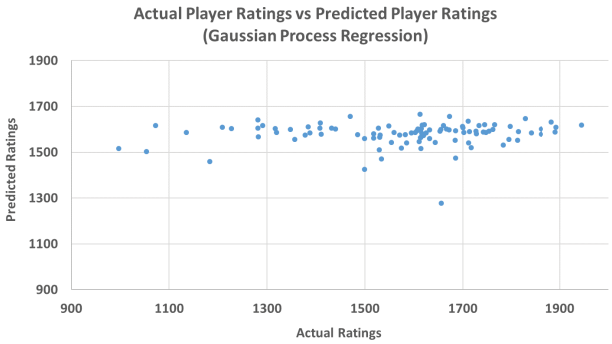

Figure 4: Scatter plot of actual and Gaussian process regression predicted ratings for players.

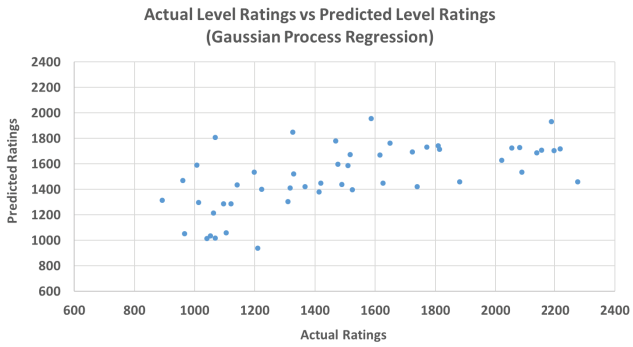

Figure 5: Scatter plot of actual and Gaussian process regression predicted ratings for levels.

To evaluate the different methods, we used leave-one-out cross validation. Scatter plots comparing the actual ratings obtained from the HIT and predicted ratings are shown in Figure 2 (players, linear regression), Figure 3 (levels, linear regression), Figure 4 (players, Gaussian process regression) and Figure 5 (levels, Gaussian process regression).

We used two approaches to compare the methods. First, we computed the root mean square error (RMSE) between the actual ratings and each of the prediction methods. These results are given in Table 1. Second, we calculated the percentage of predictions that were closer to the actual ratings than a baseline method. These results are given in Table 2 (for comparing against default) and Table 3 (for comparing against average). Overall, predictions performed relatively better for levels than for players.

We found that both linear regression and Gaussian process regression improved RMSE over using the baseline default rating—improving rating RMSE by up to 8% for players and 21% for levels. Gaussian process regression showed the greatest RMSE improvement for predicting level ratings, though it did not do much better than average for players. Average predictions actually had worse RMSE than default for levels. However, linear regression improved prediction RMSE over both baselines and thus may overall be a more consistently useful approach.

In terms of percentage of predictions that improved over baselines, both linear regression and Gaussian process regression improved a similar percentage of predictions, with Gaussian process performing slightly better than linear regression for players as compared to average predictions.

Conclusion

In this work, we examined using various methods to predict player and level ratings in a rating system used in an HCG. The use of rating systems allows a potentially unified approach to predicting both player skill and level difficulty.

We found that regression techniques improved over using the default rating; improvements were generally greater for levels than for players. This may allow accurate player and level ratings to be reached more rapidly by starting ratings out closer to their eventual values, and could have applications in improved difficulty adjustment using rating-based matchmaking. Although the predictions moved many ratings closer to their actual values, many are still far off. Thus, continuing to use the rating system during gameplay, and updating the ratings on-line based on outcomes from player attempts at levels remains useful.

In the future, this approach can be further validated on other HCGs or even entertainment games such as platformers or shooters, for example. The player features used, based on behavior in tutorials, included time spent, moves made,

and scores, which may generalize fairly easily to other games. However, level features were quite specific to *Paradox*, though some might be applicable to games with an underlying graph structure to their levels. Additionally, other features might be useful to incorporate, potentially including AI performance on levels (before they are given to players).

Beyond the scope of this current work, but additionally worth exploring, is the impact of using rating systems on player engagement in HCGs and other games (Sarkar et al. 2017). Rather than starting out players and levels with default ratings, it would be interesting to see if using features and parameters to inform initial player and level ratings as described in this work is able to more accurately assess ratings and thereby improve player engagement.

Acknowledgments

This work was supported by a Northeastern University TIER 1 grant. This material is based upon work supported by the National Science Foundation under Grant No. 1652537. We would like to thank the University of Washington's Center for Game Science for initial *Paradox* development.

References

Alexander, J. T.; Sear, J.; and Oikonomou, A. 2013. An investigation of the effects of game difficulty on player enjoyment. *Entertainment Computing* 4(1):53–62.

Ashlock, D., and Schonfeld, J. 2010. Evolution for automatic assessment of the difficulty of sokoban boards. In *IEEE Congress on Evolutionary Computation*, 1–8.

Cooper, S.; Khatib, F.; Treuille, A.; Barbero, J.; Lee, J.; Beenen, M.; Leaver-Fay, A.; Baker, D.; Popović, Z.; and Foldit Players. 2010. Predicting protein structures with a multiplayer online game. *Nature* 466(7307):756–760.

Cooper, S.; Deterding, S.; and Tsapakos, T. 2016. Player rating systems for balancing human computation games: testing the effect of bipartiteness. In *Proceedings of the 1st International Joint Conference of DiGRA and FDG*.

Csikszentmihalyi, M. 1990. *Flow: the psychology of optimal experience*. New York: Harper and Row.

Cusack, C.; Largent, J.; Alfuth, R.; and Klask, K. 2010. Online games as social-computational systems for solving NP-complete problems. *Meaningful Play*.

Dean, D.; Gaurino, S.; Eusebi, L.; Keplinger, A.; Pavlik, T.; Watro, R.; Cammarata, A.; Murray, J.; McLaughlin, K.; Cheng, J.; and Maddern, T. 2015. Lessons learned in game development for crowdsourced software formal verification. In *Proceedings of the 2015 USENIX Summit on Gaming, Games, and Gamification in Security Education*. Washington, D.C.: USENIX Association.

Denisova, A., and Cairns, P. 2015. Adaptation in digital games: the effect of challenge adjustment on player performance and experience. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '15, 97–101. New York, NY, USA: ACM.

Elo, A. E. 1978. *The rating of chessplayers, past and present*. Arco.

Engeser, S., and Rheinberg, F. 2008. Flow, performance and moderators of challenge-skill balance. *Motivation and Emotion* 32(3):158–172.

Glickman, M. E. 1999. Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 48(3):377–394.

Glickman, M. E. 2001. Dynamic paired comparison models with stochastic variances. *Journal of Applied Statistics* 28(6):673–689.

Hacker, S., and von Ahn, L. 2009. Matchin: eliciting user preferences with an online game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, 1207–1216. Boston, MA, USA: ACM.

Herbrich, R.; Minka, T.; and Graepel, T. 2007. TrueSkill(TM): a Bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, 569–576. MIT Press.

Hunicke, R. 2005. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*, ACE '05, 429–433. New York, NY, USA: ACM.

Jarušek, P., and Pelánek, R. 2010. Difficulty rating of Sokoban puzzle. In *Proceedings of the Fifth Starting AI Researchers' Symposium*, 140–150. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Kim, J. S.; Greene, M. J.; Zlateski, A.; Lee, K.; Richardson, M.; Turaga, S. C.; Purcaro, M.; Balkam, M.; Robinson, A.; Behabadi, B. F.; Campos, M.; Denk, W.; Seung, H. S.; and EyeWirers. 2014. Space-time wiring specificity supports direction selectivity in the retina. *Nature* 509(7500):331–336.

Kirkman, R. 2010. pyglicko2: a Python Implementation of the Glicko-2 algorithm.

Lomas, D.; Patel, K.; Forlizzi, J. L.; and Koedinger, K. R. 2013. Optimizing challenge in an educational game using large-scale design experiments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, 89–98. Paris, France: ACM.

Mateescu, R. 2011. Treewidth in industrial SAT benchmarks. Technical Report MSR-TR-2011-22, Microsoft Research.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, 459–465. San Jose, California: AAAI Press.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, É. 2011. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Sarkar, A.; Morrison, C.; Dorn, J. F.; Bedi, R.; Steinheimer, S.; Boisvert, J.; Burggraaff, J.; D'Souza, M.; Kontschieder, P.; Rota Bulò, S.; Walsh, L.; Kamm, C. P.; Zaykov, Y.;

Sellen, A.; and Lindley, S. 2016. Setwise comparison: consistent, scalable, continuum labels for computer vision. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, 261–271. New York, NY, USA: ACM.

Sarkar, A.; Williams, M.; Deterding, S.; and Cooper, S. 2017. Engagement effects of player rating system-based matchmaking for level ordering in human computation games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*.

Sauermann, H., and Franzoni, C. 2015. Crowd science user contribution patterns and their implications. *Proceedings of the National Academy of Sciences* 112(3):679–684.

Sinz, C. 2007. Visualizing SAT instances and runs of the DPLL algorithm. *Journal of Automated Reasoning* 39(2):219–243.

Siu, K., and Riedl, M. O. 2016. Reward systems in human computation games. In *Proceedings of the SIGCHI Annual Symposium on Computer-Human Interaction in Play*.

Sturn, T.; Wimmer, M.; Salk, C.; Perger, C.; See, L.; and Fritz, S. 2015. Cropland Capture - a game for improving global cropland maps. In *Proceedings of the 10th International Conference on the Foundations of Digital Games*.

van Kreveld, M.; Loffler, M.; and Mutser, P. 2015. Automated puzzle difficulty estimation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 415–422.

von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 319–326. Vienna, Austria: ACM.