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

Creating a Hyper-Agent for Solving Angry Birds Levels

Matthew Stephenson, Jochen Renz
Research School of Computer Science
Australian National University
Canberra, Australia
matthew.stephenson@anu.edu.au, jochen.renz@anu.edu.au

Abstract

Over the past few years the Angry Birds AI competition has been held in an attempt to develop intelligent agents that can successfully and efficiently solve levels for the video game Angry Birds. Many different agents and strategies have been developed to solve the complex and challenging physical reasoning problems associated with such a game. However, the performance of these various agents is non-transitive and varies significantly across different levels. No single agent dominates all situations presented, indicating that different procedures are better at solving certain levels than others. We therefore propose the construction of a hyper-agent that selects from a portfolio of sub-agents whichever it believes is best at solving any given level. This hyper-agent utilises key features that can be observed about a level to rank the available candidate algorithms based on their expected score. The proposed method exhibits a significant increase in performance over the individual sub-agents, and demonstrates the potential of using such an approach to solve other physics-based games or problems.

Introduction

The creation of an intelligent agent that can reason and predict the outcome of actions in a physical simulation environment, typically with inaccurate information, is a key subject of investigation in the field of AI. It is particularly important for the development of such agents to integrate the areas of computer vision, machine learning, knowledge representation and reasoning, planning, and reasoning under uncertainty. The Angry Birds AI (AIBirds) competition was created as a means to promote the research and creation of these agents through the use of the physics-based simulation game Angry Birds (Renz 2015). This type of physical reasoning problem is very different to traditional games as the attributes and parameters of various objects are often imprecise or unknown, meaning that it is very difficult to accurately predict the outcome of any action taken (Renz et al. 2016). Many of the previous agents that have participated in this competition employ a variety of techniques, including qualitative reasoning (Waga, Zawidzki, and Lechowski 2016), internal simulation analysis (Polceanu and Buche 2013; Schiffer, Jourenko, and Lakemeyer 2016), logic programming (Calimeri et al. 2016), heuristics (Dasgupta et al. 2016), Bayesian inferences (Tziortziotis, Papagiannis, and Blekas 2016; Narayan-Chen, Xu, and Shavlik 2013), and structural analysis (Zhang and Renz 2014). However, none of these agents has ever come close to being the dominant performer across all levels (AIBirds 2017), indicating that these methods are best suited to specific situations.

The fact that different agents perform better at different levels suggests that the construction of a "hyper-agent" (a.k.a. portfolio agent or ensemble agent), which selects from a portfolio of various sub-agents, would be able to utilise the combined strengths of their techniques (Mendes, Togelius, and Nealen 2016). This is an idea that has been suggested previously under the terms hyper-heuristic (Burke et al. 2013; 2010) and algorithm selection (Kotthoff 2014). The hyper-agent proposed in this paper uses a set of training levels to acquire information about how particular features of a level relate to each sub-agent's performance. A prediction model is then created that allows the hyper-agent to determine which sub-agent(s) would likely be the most successful for any unknown levels it encounters (i.e. an offline learning approach). Hyper-agents have been proposed previously for domains such as task scheduling (Cowling, Kendall, and Soubeiga 2001), packing problems (López-Camacho et al. 2014) and examination timetabling (Burke et al. 2012), as well as for multiple video game genres including strategy games (Li and Kendall 2017), card games (Elyasaf, Hauptman, and Sipper 2012), puzzle games (Salcedo-Sanz et al. 2014) and General Video Games (GVGAI) (Bontrager et al. 2016; Horn et al. 2016; Mendes, Togelius, and Nealen 2016).

Whilst some of the agents that have previously participated in the AIBirds competition have utilised various simple strategies based on level properties before, none have yet combined different fields of AI based on higher level features. Some of these approaches are faster, whilst others may be more consistent, or adaptable to new scenarios. Physical simulation games such as Angry Birds provide a large and varied range of creative and challenging levels that cannot yet be solved by a single AI technique, despite the fact that people and even children are able to solve most of these levels relatively quickly and easily (Renz et al. 2015). Combining these techniques therefore seems to be the most

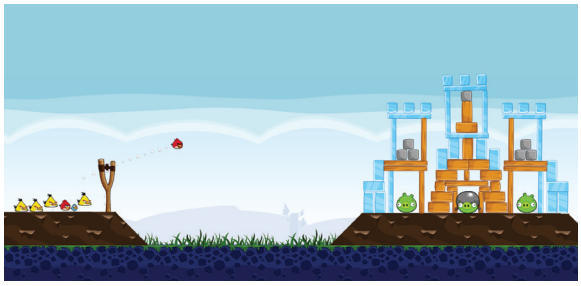

Figure 1: Screenshot of a level from the Angry Birds game.

effective and promising means of developing a successful AI agent for both Angry Birds and other real-world physics problems.

Background

Angry Birds Game

Angry Birds is a popular physics-based puzzle game in which the player uses a slingshot to shoot birds at pigs, with structures composed of blocks and other physical objects protecting them, see Figure 1. The goal of each level is to kill all pigs using a set number of birds provided. All objects within the level have properties such as location, size, mass, friction, density, etc., and obey simplified physics principles defined within the game's engine. Blocks are also made of one of three materials, wood, stone or ice. Different bird types are available with different properties, and pigs are killed once they take enough damage from either the birds directly or by being hit with another object. The player can choose the angle and speed with which to fire a bird from the slingshot, as well as a tap time for when to activate the bird's special ability if it has one, but cannot alter the ordering of the birds or affect the level in any other way. The difficulty of this game comes from predicting the physical consequences of actions taken, and accurately planning a sequence of shots that will result in success. Points are awarded to the player once the level is solved based on the number of birds remaining and the total amount of damage caused.

AIBirds Competition

In this competition, agents are tasked with playing a set number of unknown Angry Birds levels within a given time, attempting to score as many points as possible in each level. The exact location and parameters of certain objects, as well as the current internal state of the game, are not directly accessible. Instead, information about the level is provided using a computer vision module, effectively meaning that an agent gets exactly that same input as a human player. Agents are required to solve these levels in real-time and can attempt levels in any order and as many times as they like. Once the time limit has expired the maximum scores that an agent achieved for each level are summed up to give its final score. Agents are then ranked based on this value and after several rounds of elimination a winner is declared. The eventual goal of this competition is to design AI agents that can play new levels as well as or better than human players.

Agent Discussion

Our proposed hyper-agent selects from a portfolio consisting of the eight agents that participated in the 2016 AIBirds competition. Whilst there have been over 30 different agents that have participated in the AIBirds competition over the years, the agents from the latest competition represent the best that are currently available. A brief description of each of these agents is given below, with full details available on the AIBirds website (AIBirds 2017).

2016 Competition Past Agents

Naive Agent The Naive agent is provided to all competition entrants as a useful starting point upon which to create their own AI agent. It fires the currently selected bird at a randomly chosen pig using either a low or high trajectory (also chosen at random). No other objects apart from the current bird and pigs are used when determining a suitable shot, and tap times are fixed for each bird based on the total length of its trajectory.

Datalab Agent The Datalab agent uses a combination of four different strategies when attempting to solve a level. These can be described as the destroy pigs, building, dynamite and round blocks strategies. The decision of which strategy to use is based on the environment, possible trajectories, currently selected bird and remaining birds. The destroy pigs strategy attempts to find a trajectory that intersects with as many pigs as possible. The building strategy identifies and targets groups of connected blocks that either protect pigs or are near to them. The dynamite strategy ranks each TNT box within the level based on the number of pigs, stone blocks and other TNT boxes that are nearby. The round blocks strategy attempts to either hit round blocks directly or else destroy objects that are supporting round blocks.

IHSEV Agent The IHSEV agent creates an internal Box2D simulation of the level, within which it tries out many shot angles and tap times. The shot that destroys the most pigs is always selected. However, the simulation is not a perfect representation of the environment. The agent does not use any information about the number or type of remaining birds when deciding which shot to take. A future plan to adapt the agent's environmental simulation based on the deviation between the actual and expected outcome of a shot was proposed but has not yet been implemented.

Angry-HEX Agent The Angry-HEX agent uses HEX programs to deal with decisions and reasoning, while the computations are performed by traditional programming. HEX programs are an extension of answer set programming (ASP) which use declarative knowledge bases for information representation and reasoning. The Reasoner module of this agent determines several possible shots based on different strategies. These shots are then simulated using a Box2D simulation, with the shot that kills the most pigs being selected as the ideal action (number of destroyed blocks being used as a tiebreaker). The trajectory module of the base program was improved to take the thickness of the currently selected bird into account, as well as the ability to select several different points on a block as the target location.

Eagle's Wing Agent The Eagle's Wing agent chooses from five different strategies when deciding what shot to perform. These are defined as the pigshooter, TNT, most blocks, high round objects and bottom building blocks strategies. The pigshooter strategy attempts to find a trajectory that either targets an unprotected pig, or includes multiple pigs within it. The TNT strategy aims for any TNT box that can cause significant damage to a large region. The many blocks strategy finds the trajectory that destroys the most blocks (highly dependent on the type of bird being used). The high round objects strategy attempts to destroy objects close to large round objects that are high above the ground. The bottom building block strategy targets blocks that are important to a structure's overall stability.

SEABirds Agent The SEABirds agent uses an Analytic Hierarchy Process (AHP) for deciding which shots to make, and determines the best object or structure to hit based on five different criteria. This includes the Y-axis position, surrounding objects/structures, breakability (for currently selected bird type), relative distance to pigs and whether the object is a TNT box. The relative importance of each criteria compared to the other alternative options is calculated using a collection of training levels.

s-birds Agent The s-birds agent has two different approaches for determining the most effective shot to perform. The first strategy is called the bottom-up approach and identifies a set of candidate target blocks for the level based on the potential number of affected pigs. The second strategy is called the top-down approach and utilizes the crushing/rolling effect of a bird or round block onto pigs, as well as the toppling effect of thinner blocks. Suitable target blocks are ranked based on the expected number of pigs killed and the likelihood of the shot's success. The penetration factor of specific bird types against certain materials is also considered when determining if a block can be hit.

Bambirds Agent The Bambirds agent creates a qualitative representation of the level and then chooses one of nine different strategies based on its current state. This includes approaches such as utilizing blocks within the level to create a domino effect, targeting blocks that support heavy objects, maximum structure penetration and prioritizing protective blocks, as well as simpler options such as targeting pigs/TNT or utilizing certain bird's special abilities. These strategies are each given a score based on their estimated damage potential for the current bird type. A strategy is then chosen randomly, with this score being used to determine the likelihood of selection (i.e. shots that are believed to be the most effective are more likely to be chosen).

Meta Strategies

Whilst the techniques each agent uses for solving a level have been discussed, many agents also feature a variety of different strategies for determining which levels are to be played. The time given to each agent in the AIBirds competition is typically high enough that it can attempt each level multiple times. Some agents choose to attempt all levels once before replaying any unsolved levels (such as Datalab and Eagle's Wing) whilst others attempt a level multiple times before moving on (such as s-birds and SEABirds). Angry-HEX and Bambirds are also able to remember the shots and strategies previously carried out, to aid them when re-attempting levels later on. Whilst most agents try to solve all levels before re-attempting those already solved, Bambirds calculates a probability of attempting each level based on an estimated number of points for solving it, the number of times it has been played and the current score for that level. For our hyper-agent we will use the following simple meta-strategy. All levels are to be attempted at least once, after which all still unsolved levels are repeatedly played again. If all levels are solved then we simply cycle through all the available levels.

Methodology

This section provides an overview of the methods used to create the proposed hyper-agent. This involves both the collection of important level features which can be used to create models for predicting each agent's score, as well as how the hyper-agent uses these score prediction models to choose a sub-agent from its available portfolio when attempting to solve an unknown level.

Feature Collection

Identifying features of a level that influence agent performance is one of the most important aspects in the creation of a hyper-agent. For this type of game, the two factors that make up an agent's performance are the score it achieves for a level and how long it took it to achieve that score. After analysing the strategies and approaches of our sub-agents we defined a list of 24 different numerical properties of a level which we believe may influence the performance of certain agents, see Table 1. These include basic level features such as the number and type of different birds or blocks within the level, as well as more complex attributes such as the number of block connections or the overall dispersion of pig locations. The values for each of these properties are calculated using the information provided by the competition software's computer vision module. These values may therefore be subject to noise or other imperfections which will affect the reliability of the information perceived. However, this is the same error that an agent would have to face whilst playing unknown levels in real time. The first time the proposed hyper-agent plays a level it will first calculate the values for each of these features, after which it will select an appropriate sub-agent from its portfolio. The time required to calculate these features is very short, taking less than a few seconds after the level has loaded.

Agent Selection

Using the collected features of certain levels, along with each agent's average score at those same levels, we can construct machine learning models to predict each agent's score for an unknown level based on its features. Our hyper-agent can then use these models to calculate an expected score for each sub-agent in its portfolio. This enables us to rank the agents based on their expected performance. Each time

Feature	Description
#Pigs	Number of pigs
#Wood	Number of wood blocks
#Stone	Number of stone blocks
#Ice	Number of ice blocks
AreaWood	Total area of wood blocks
AreaStone	Total area of stone blocks
AreaIce	Total area of ice blocks
#RedBirds	Number of red birds
#BlueBirds	Number of blue birds
#YellowBirds	Number of yellow birds
#BlackBirds	Number of black birds
#WhiteBirds	Number of white birds
#TNT	Number of TNT boxes
#Round	Number of round blocks
AreaTerrain	Total area of static terrain
#BelowRound	Number of pigs below round blocks
#Blocked	Number of pigs that have terrain blocking the player's shot trajectory to them
#Reachable	Number of pigs that can be hit directly by the player (no protection)
#OutOfRange	Number of pigs beyond the range of the player's shots
LevelWidth	Width of the level
LevelHeight	Height of the level
PigDispersion	Overall dispersion of pig locations, calcuated using method proposed in (Stephenson and Renz 2016a)
AvgAspectRatio	Average aspect ratio of all blocks
#BlockConnections	Number of edges where blocks touch

Table 1: Selected level features to model agent performance

the hyper-agent attempts a level it will choose the highest ranked agent that has not already been tried. If all agents have played a level then we repeat this selection process, starting again from the highest ranked agent.

Experiments and Results

In order to fully create and test our hyper-agent we require three distinct steps. First, we need to use a set of training levels to evaluate each sub-agent's performance based on the features within those levels. Second, we need to use this performance information to construct score prediction models for each sub-agent that can be used on unknown levels. Third, we will use a new set of testing levels to compare the performance of our hyper-agent against that of each of the original sub-agents.

A total of 105 levels are available from the "Poached Eggs" and "Mighty Hoax" episodes of the original Angry Birds game. Other levels from different Angry Birds games or episodes feature objects that are not detectable by the vision module and are thus not usable by any of the Angry Birds agents which are currently available. Of these 105 levels we found that six of them caused issues with the vision module, preventing certain key objects from being identified. This reduced the total number of viable levels with which to train our hyper-agent to 99. We also have a collection of 80 new levels that were featured in the three previous AIBirds competitions. These were not used in the training process but were instead used to test the hyper-agent and evaluate its performance. The experiments conducted were all carried out on an Ubuntu(16.04) 64-bit desktop PC with an i7-4790 CPU and 16GB RAM.

Agent	Average Score	Average Shot Time
Naive	17570	31.74
Datalab	37557	19.82
IHSEV	24402	31.86
Angry-HEX	21253	24.19
Eagle's Wing	36468	20.95
SEABirds	29978	35.01
s-birds	19628	55.64
Bambirds	23960	28.27

Table 2: Agent performance on training levels

Agent Performance

Using our collection of 99 training levels, we tasked each sub-agent with solving a level with the highest score possible within ten minutes. As some agents use their past attempts to tailor their future ones, we treated each new attempt like a brand-new level. This process was repeated five times, to give five rounds of ten minutes, within which each agent attempted to score as many points as possible. The maximum score from each round was recorded for each agent and the average of these scores across all rounds gave the agent's final score for that level. This process was followed so as to better suit each agent's overall performance in the AIBirds competition environment, where an agent is given a fixed amount of time to solve a collection of levels rather than a set number of attempts.

The average score for each agent across all levels, along with the average time in seconds that each agent took to make a single shot, is provided in Table 2. Out of the 99 levels used, only five of them could not be successfully completed by any agent, giving us a total of 94 completed levels with which to build our score prediction models. From this information we can see a large disparity in the average scores of the best agent (Datalab) and the worst agent (Naive) of almost 20,000 points, but also that there is a reasonably gradual increase from the worst to best agent, with the jumps in agent's scores never being greater than 6500 points. There is also a moderate negative correlation (coefficient of -0.458) between the average score and shot time for each agent, indicating that having a faster shot time typically leads to a greater overall score, which is likely due to the increased number of level attempts this results in.

Prediction Model Comparison

Using the agent scores from the training levels, along with the features recorded using the computer vision module, we can create a score prediction model for each agent. However, this prediction model could be created with one of multiple machine learning techniques. The Weka machine learning software (Hall et al. 2009) provides several ready-made algorithms for this purpose. Possible popular

options include using Linear Regressions, Multi-Layer Perceptrons (MLP), Support Vector Machines (SMOreg), k-Nearest Neighbours (IBk), Random Trees, Random Forests, and M5 Trees (M5P). Each of these methods can be used to create a regression model to predict an agent's expected score for an unknown level. However, the accuracy of the models created by each technique differs from agent to agent, making choosing the right model for each agent extremely important.

In order to compare the models created by each method for each agent, we performed 10-fold cross validation on each model. The mean absolute error for each model was recorded, allowing us to determine the best score prediction model for each agent. The results of this analysis can be seen in Table 3, with the lowest error values for each agent given in bold (Note. the results for MLP, M5P and Random trees were excluded to save space). From this we can see that Random Forests and k-Nearest Neighbours (k=5) are best at representing three agents each, while Linear regression and SMOreg best model one agent each. MLP, M5P and Random Trees did not best represent any agents. The best performing model for each agent was selected to be used on unknown levels. Comparing these errors against the average scores acheived by each agent gives a mean error of 72.1% over all eight methods. Whilst this may seem like a fairly low signal-to-noise ratio the mean error for simply predicting the average performance of each agent (ZeroR) is 88.5%, indicating that there is an extremely large amount of variation in agent scores between levels.

Agent	Linear Regr	SMOreg	Random Forest	IBk (k=5)
Naive	**16360**	16450	16969	16414
Datalab	19700	21925	**18838**	20995
IHSEV	20251	19225	19763	**18135**
Angry-HEX	21216	22060	22060	**18921**
Eagle's Wing	20548	20384	**18472**	19761
SEABirds	27594	28508	**21842**	22012
s-birds	17529	**17191**	18682	18174
Bambirds	15407	15917	14201	**14083**

Table 3: Mean absolute error for score prediction models

Hyper-Agent Analysis

Using the agent selection method previously described, we are now able to create a hyper-agent to play unknown levels of Angry Birds. Using the 80 levels featured in the past three years of AIBirds competitions, we can compare our new hyper-agent against the eight original agents which make up its portfolio. Using the same rules as in the AIBirds competition, each agent is tasked with playing a collection of eight levels in 30 minutes (one round of the competition) with the combined maximum score achieved for each level making up an agent's total score for that round. After playing all ten rounds of eight levels, we can then compare each agent's overall performance, see Table 4.

From this we can see our hyper-agent performed better overall than any other single agent, both in terms of score and the number of levels solved. Out of the ten rounds played, our hyper-agent came first in seven of them, with it coming second in the other three to SEABirds in the qualification round, Datalab in the 2014 semi-finals round and IHSEV in the 2016 quarter-finals round. Using these scores, we can determine that had our hyper-agent competed against these agents in the last three AIBirds competitions, and performed the same as in these tests, then it would have won all of the competitions from those years.

The distributions of each agent's scores were also compared by performing a Mann-Whitney-Wilcoxon (MWW) test, in order to determine whether or not the hyper-agent's performance statistically differs from that of the other agents (Fay and Proschann 2010). The bottom row of Table 4 shows the P-values for each sub-agent's score distributions when compared against the hyper-agent. Using a p-value of less than 0.05 as a marker for significance, this test demonstrates that for all agents, with the exception of Datalab, we can reject the null hypothesis that the difference in these scores is due to random sampling.

To ensure that this improved score was not simply due to the increased number of different agents attempting each level, we also ran two naive hyper-agents on the competition levels. The first naive hyper-agent selects a sub-agent based only on the average performance of each agent from the training levels (does not observe anything about the level's features), whilst the second randomly selects one of the eight available agents each time it attempts a level. These naive hyper-agents gave total level scores of 3783850 and 3176200 respectively. We also tried the randomly selecting hyper-agent again, but this time with only the top four agents (Datalab, Eagle's Wing, SEABirds and IHSEV) being used in the selection pool. Whilst this increased its total score to 3697150, its performance is still well below that of our proposed hyper-agent.

Discussion

The proposed hyper-agent uses an assortment of score prediction models to rank the sub-agents available in its portfolio based on a given level's features. These models were created using one of several possible machine learning techniques, with different techniques being used to create models for different sub-agents. The use of different model designs makes it difficult to directly compare which features most affected each sub-agent's performance, and in addition, the sheer number of features makes for an extremely detailed and complex comparison. Nevertheless, we will briefly mention some noticeable points of interest.

Whilst the effects of many of the more common and fluctuating level properties, such as the number/area of certain block materials and the level's width/height, varied greatly from agent to agent in terms of importance, there were several features that seemed to be universally good or bad for most agents. For example, #BlackBirds had a very positive affect on the predicted score for all eight agents, likely due to the large amount of damage this bird type causes and their simplicity of use. Additional factors such as AvgAspectRatio also had a large positive affect on most of the higher ranked agents, whilst PigDispersion and PigsBlocked had a

Round	Naive	Datalab	IHSEV	Angry-HEX	Eagle's Wing	SEABirds	s-birds	Bambirds	Hyper
Qualification	251360	349640	195350	237420	384370	**397810**	143880	272670	397600
Quarter 2014	187180	309920	109920	134840	282110	319730	227400	161150	**332270**
Semi 2014	400980	**586120**	439520	402270	442800	453730	142760	383380	524400
Final 2014	209130	243160	257410	130630	250970	55800	0	132400	**338330**
Quarter 2015	68020	286450	163790	229090	346760	97370	84300	161580	**351300**
Semi 2015	145910	300330	166750	64480	299220	282600	151800	103950	**375670**
Final 2015	131660	440680	458030	462600	191970	383750	337970	417460	**483610**
Quarter 2016	251080	327490	**444560**	231300	252100	328570	182070	280930	336840
Semi 2016	436870	371100	562820	475840	420170	293410	190840	406200	**610280**
Final 2016	390050	415320	288720	347960	421790	385740	356050	426980	**469960**
Total Score	2472240	3630210	3086870	2716430	3292260	2998510	1817070	2746700	**4220260**
Levels Solved	37	59	52	38	52	49	27	42	**69**
MWW Score	0.0001	0.2627	0.0091	0.0014	0.0477	0.0067	0.0000	0.0008	N/A

Table 4: Agent performance on AIBirds competition levels

strong negative effect on the lower ranked agents. The only feature that greatly affected most agents predicted scores but in opposite directions was #WhiteBirds, which had a positive effect on the Datalab, SEABirds and Eagle's Wing agents, but a negative effect on all the others. This is likely due to the fact that using the white bird effectively is very difficult, so less skilled agents cannot usually complete levels that contain them and so receive zero points.

Whilst the proposed hyper-agent performs better than each of the individual sub-agents, it still has several limitations that could be addressed to improve its performance further. The main benefit of the proposed hyper-agent is its ability to use the multiple AI techniques employed by its portfolio of agents. However, some of these agents are more similar in their approach than others. It may be possible that there are correlations between certain sub-agents and the levels which they can solve, meaning that levels which cannot be solved by one of these agents would also probably not be solved by the other. Taking this into account could allow us to update each agent's expected score based on which agents have already attempted the level.

Another improvement that could increase the hyper-agent's overall performance would be to design a more complex meta-strategy, which uses the predicted score of each agent to identify the levels that will net the most points if solved. Resetting a level halfway through an attempt to try another agent, if the hyper-agent believes that the current agent can no longer solve the level, may also be an interesting topic of investigation. A greater understanding of why certain sub-agents perform better at certain levels would also help create better score prediction models.

Increasing the number of levels that are available for training the hyper-agent would naturally increase the accuracy of its predictions. Whilst there are a reasonably large number of Angry Birds levels available, most of them contain objects that are not yet incorporated into the AIBirds competition framework, and are thus not recognised by any of the current agents. A possible solution to this problem would be to utilise a level generator to create new levels with which to train our hyper-agent. Several Angry Birds level generators have been proposed previously (Ferreira and Toledo 2014; Stephenson and Renz 2016b; Pereira et al. 2016) and provide the potential to build much more accurate models. We could also utilise algorithms that can identify new Angry Birds objects (Ge, Renz, and Zhang 2016).

Conclusion

This paper has presented an approach to creating a hyper-agent for Angry Birds that selects from a portfolio of other prior agents. Using a set of training levels, we were able to extract features that may be deemed relevant to an agent's overall performance and use this to train a set of regression models which predict each sub-agent's expected score. When confronted with an unknown level we can use these score prediction models, along with the level's features, to rank each of the available sub-agents. Our proposed hyper-agent can then use this ranking to determine the order in which to use the sub-agents available.

Comparing the scores of our hyper-agent against its individual constituent agents, for the levels used for that past three AIBirds competitions, revealed that it performed considerably better than all of them. This encouraging result demonstrates the potential of hyper-agents which utilise multiple AI techniques, not just for Angry Birds but for physics-based problems in general. We have also discussed many possible areas for improvement, which may further increase the performance of our hyper-agent.

In the future, we aim to not only increase the abilities of the current hyper-agent but to also further explore the domains in which our methods could be applied. As an example, the general video game AI competition (GVGAI) has recently revealed a collection of physics-based games which it intends to add to its current line-up. This is a promising new area within which to create a hyper-agent and would pose many interesting challenges and opportunities for further research. Additional areas of AI such as content generation are also possible applications, where score prediction models from hyper-agents could be used to help estimate the difficulty of a generated level.

References

AIBirds. 2017. AIBirds homepage. https://aibirds.org. Accessed: 2017-04-21.

Bontrager, P.; Khalifa, A.; Mendes, A.; and Togelius, J. 2016. Matching games and algorithms for general video game playing. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 122128.

Burke, E. K.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; and Woodward, J. R. 2010. *A Classification of Hyper-heuristic Approaches*. Boston, MA: Springer US. 449–468.

Burke, E. K.; Kendall, G.; Mısır, M.; and Özcan, E. 2012. Monte Carlo hyper-heuristics for examination timetabling. *Annals of Operations Research* 196(1):73–90.

Burke, E. K.; Gendreau, M.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; and Qu, R. 2013. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society* 64(12):1695–1724.

Calimeri, F.; Fink, M.; Germano, S.; Humenberger, A.; Ianni, G.; Redl, C.; Stepanova, D.; Tucci, A.; and Wimmer, A. 2016. Angry-HEX: An artificial player for Angry Birds based on declarative knowledge bases. *IEEE Transactions on Computational Intelligence and AI in Games* 8(2):128–139.

Cowling, P.; Kendall, G.; and Soubeiga, E. 2001. *A Hyper-heuristic Approach to Scheduling a Sales Summit*. Berlin, Heidelberg: Springer Berlin Heidelberg. 176–190.

Dasgupta, S.; Vaghela, S.; Modi, V.; and Kanakia, H. 2016. s-Birds Avengers: A dynamic heuristic engine-based agent for the Angry Birds problem. *IEEE Transactions on Computational Intelligence and AI in Games* 8(2):140–151.

Elyasaf, A.; Hauptman, A.; and Sipper, M. 2012. Evolutionary design of FreeCell solvers. *IEEE Transactions on Computational Intelligence and AI in Games* 4(4):270–281.

Fay, M. P., and Proschann, M. A. 2010. WilcoxonMannWhitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics Surveys* 4:1–39.

Ferreira, L., and Toledo, C. 2014. A search-based approach for generating Angry Birds levels. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, 1–8.

Ge, X.; Renz, J.; and Zhang, P. 2016. Visual detection of unknown objects in video games using qualitative stability analysis. *IEEE Transactions on Computational Intelligence and AI in Games* 8(2):166–177.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.* 11(1):10–18.

Horn, H.; Volz, V.; Pérez-Liébana, D.; and Preuss, M. 2016. MCTS/EA hybrid GVGAI players and game difficulty estimation. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.

Kotthoff, L. 2014. Algorithm selection for combinatorial search problems: A survey. *AI Magazine* 35(3):48–60.

Li, J., and Kendall, G. 2017. A hyperheuristic methodology to generate adaptive strategies for games. *IEEE Transactions on Computational Intelligence and AI in Games* 9(1):1–10.

López-Camacho, E.; Terashima-Marin, H.; Ross, P.; and Ochoa, G. 2014. A unified hyper-heuristic framework for solving bin packing problems. *Expert Systems with Applications* 41(15):6876 – 6889.

Mendes, A.; Togelius, J.; and Nealen, A. 2016. Hyper-heuristic general video game playing. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.

Narayan-Chen, A.; Xu, L.; and Shavlik, J. 2013. An empirical evaluation of machine learning approaches for Angry Birds. In *IJCAI Symposium on AI in Angry Birds*.

Pereira, L. T.; Toledo, C.; Ferreira, L. N.; and Lelis, L. H. S. 2016. Learning to speed up evolutionary content generation in physics-based puzzle games. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, 901–907.

Polceanu, M., and Buche, C. 2013. Towards a theory-of-mind-inspired generic decision-making framework. In *IJCAI Symposium on AI in Angry Birds*.

Renz, J.; Ge, X.; Gould, S.; and Zhang, P. 2015. The Angry Birds AI competition. *AI Magazine* 36(2):85–87.

Renz, J.; Ge, X.; Verma, R.; and Zhang, P. 2016. Angry Birds as a challenge for artificial intelligence. In *AAAI Conference on Artificial Intelligence*, 4338–4339.

Renz, J. 2015. AIBIRDS: The Angry Birds artificial intelligence competition. In *AAAI Conference on Artificial Intelligence*, 4326–4327.

Salcedo-Sanz, S.; Matías-Román, J. M.; Jiménez-Fernández, S.; Portilla-Figueras, A.; and Cuadra, L. 2014. An evolutionary-based hyper-heuristic approach for the Jawbreaker puzzle. *Applied Intelligence* 40(3):404–414.

Schiffer, S.; Jourenko, M.; and Lakemeyer, G. 2016. Akbaba: An agent for the Angry Birds AI challenge based on search and simulation. *IEEE Transactions on Computational Intelligence and AI in Games* 8(2):116–127.

Stephenson, M., and Renz, J. 2016a. Procedural generation of complex stable structures for Angry Birds levels. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.

Stephenson, M., and Renz, J. 2016b. Procedural generation of levels for Angry Birds style physics games. In *Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-16)*, 225–231.

Tziortziotis, N.; Papagiannis, G.; and Blekas, K. 2016. A bayesian ensemble regression framework on the Angry Birds game. *IEEE Transactions on Computational Intelligence and AI in Games* 8(2):104–115.

Waga, P. A.; Zawidzki, M.; and Lechowski, T. 2016. Qualitative physics in Angry Birds. *IEEE Transactions on Computational Intelligence and AI in Games* 8(2):152–165.

Zhang, P., and Renz, J. 2014. Qualitative spatial representation and reasoning in Angry Birds: The extended rectangle algebra. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, KR'14, 378–387.