A CBR Approach to the Angry Birds Game

Adil Paul and Eyke Hüllermeier

Department of Computer Science
University of Paderborn, Germany
{adil.paul,eyke}@upb.de

Abstract. In this paper, we present a CBR approach for implementing an agent playing the well-known Angry Birds game. We adopt a preference-based procedure for constructing the case base, collecting experience from a random agent that continually explores the problem-solution space and improves the quality of already found solutions. As the retrieve phase involves finding a game scene similar to a given one, we develop a measure to assess the dissimilarity between two game scenes, which is based on solving appropriate linear assignment problems. A comparison of our agent with state-of-the-art computer programs shows promising results.

1 Introduction

Angry Birds is a popular video game, in which the player has to shoot birds from a slingshot at pigs that are protected with objects from different types of materials, including wood, stone, and ice. Some birds have specific capabilities that allow them to explode, split into several birds, pick up speed, etc. The game has different levels, each level coming with its specific representation of pigs and objects hiding them. A level is solved when all the pigs are destroyed, and the goal of a player is to solve all the levels, keeping the number of shot birds as low as possible.

Since the first edition of the Angry Birds AI competition in 2012, different approaches, ranging from qualitative representation and reasoning over simulation of game scenes to classical supervised machine learning algorithms, have been leveraged to build agents playing the game. In this paper, we develop an Angry Birds agent on the basis of the case-based reasoning (CBR) paradigm. To the best of our knowledge, this is the first CBR approach to Angry Birds. One of the main components of our Angry Birds agent is a case base that stores problem-solution pairs, i.e., game scenes and appropriate best shots. We use a preference-based approach to build the case base, which compares different solutions for a given problem and maintains the better one.

The rest of the paper is organized as follows. In the next section, we briefly review some of the existing approaches for agents playing the Angry Birds game. In Section 3, we present our approach, and in Section 4, we analyze its performance experimentally. We conclude our work and outline possible directions for future work in Section 5.

2 Existing Approaches

Most of the work so far has been concerned with the representation of the different types of objects in Angry Birds. Lin et al. [7] classify the objects into dynamic, which are mainly convex polygons, and static ones, which comprise concave polygons, and use bounding convex polygons (BCPs) to represent the former and edge detection and Hough transform to detect the latter. Zhang and Renz [12, 13] also make use of the spatial representation of objects and, moreover, reason about their stability. They build on an extension of the rectangle algebra to assess the stability of blocks of objects, upon which they can decide where to hit a block so as to affect it maximally.

In [11], the authors assign a numerical score to each reachable object, based on its physical properties. The score is supposed to reflect the extent of damage it suffers if being hit, and shoots at objects with low stability but high influence on pigs or shelters of pigs. Ferreira et al. [3] also assign a utility value to the objects based on spatial properties, but because of the lack of certainty in the position of the objects, they incorporate concepts of probability and uncertainty to determine the chance of a bird to hit a given target.

Simulation-based approaches include the work by Polceanu and Buche [9], who build their decision making based on the theory of mental simulation. More precisely, their agent observes the effects of performing multiple simulations of different shots in a given game scene and selects the optimal solution based on these results.

The remaining category of approaches encompasses agents that leverage different machine learning algorithms. In order to learn how to judge shots, Narayan-Chen et al. [8] train a weighted majority and a Naive Bayes algorithm on a data set consisting of good and bad shots in different states of the game. Tziortziotis and Buche [10] use a tree structure to represent the objects in a game scene, and formulate the problem of selecting an object for shooting as a regression problem. They associate with each pair of object material and bird a Bayesian linear regression model, building a competitive ensemble of models, whose parameters are estimated in an online fashion. The decision is then made according to the best prediction of the ensemble model.

3 A Case-based Angry Birds Agent

We employ the CBR approach [1] to build an agent that plays the Angry Birds game. The experience-oriented learning and reasoning paradigm of CBR first of all requires the creation of a case base that stores problem-solution pairs. As the problem space in the domain of Angry Birds is infinite, and no exact characterization of an optimal solution (the best shot) for a problem (a description of a game scene) exists, a way of gathering expressive pairs of problems and approximate solutions (game scenes together with reasonably good shots) is needed. Further, a game scene in Angry Birds comprises objects with different shapes, which should be represented and stored appropriately. Thus, a representation that reflects the spatial properties of the different objects involved in the game is another concern. Lastly, once the case base is built and appropriately stored, the problem of retrieving cases similar to a given query case needs to be addressed, which in turn necessitates assessing the similarity between two game scenes. In the following, we elaborate on each of these issues.

3.1 Case Base Construction

The core of a CBR system is a case base that stores previously encountered problems and associated solutions. In the context of Angry Birds, a single case should enclose a problem description part, with a representation of a game scene, covering the slingshot and all objects and pigs, and a solution part, containing the best shot one can execute in the given scene. The notion of an optimal solution in a given game scene, i.e., the shot that will lead to the highest change in score, is actually not well-defined. Therefore, we need a procedure to find solutions of at least close-to-optimal quality.

Inspired by the general framework of preference-based CBR [5], we construct a case base by comparing the quality of solutions that have been tried so far. The basic principle of the approach consists of randomly trying different solutions for a problem and maintaining the best one. The advantages of this approach are two-fold. First, because of its self-adaptive nature, it does not rely on any external domain expert to provide solutions for the potentially infinite number of problems. Second, as the problem and solution space are explored more and more, the extent of the case base is enlarged and its quality is improved over time.

In the context of Angry Birds, we concretise the approach as follows. We let arbitrary agents play in different game scenes and record the game scene along with the shot executed by the agent and the change in score. Once we encounter a game scene which is similar to another one already contained in the case base, and where the agent performs better, we replace the solution part of the old case (i.e., the shot) with the new one. The steps of the process of case base construction are outlined in Figure 1 as a flowchart diagram.

3.2 Case Representation

The Angry Birds game involves different types of objects: a sling, hills, pigs, blocks of stone, wood or ice, TNTs and birds with different capabilities expressed in terms of colours, including red, yellow, blue, black, and white. The Angry Birds Basic Game Playing Software [4] provides two possibilities of representing theses objects: the Minimum Bounding Rectangle (MBR) and the real shape representation. While the MBR segmentation of an object consists solely of finding a rectangle with minimal area, which completely covers it, the real shape segmentation represents the objects more precisely using circles, rectangles and polygons, and distinguishes between hollow and solid objects. As such, the latter is more precise but also more costly to compute. In this paper, we confine ourselves to the MBR representation of objects.

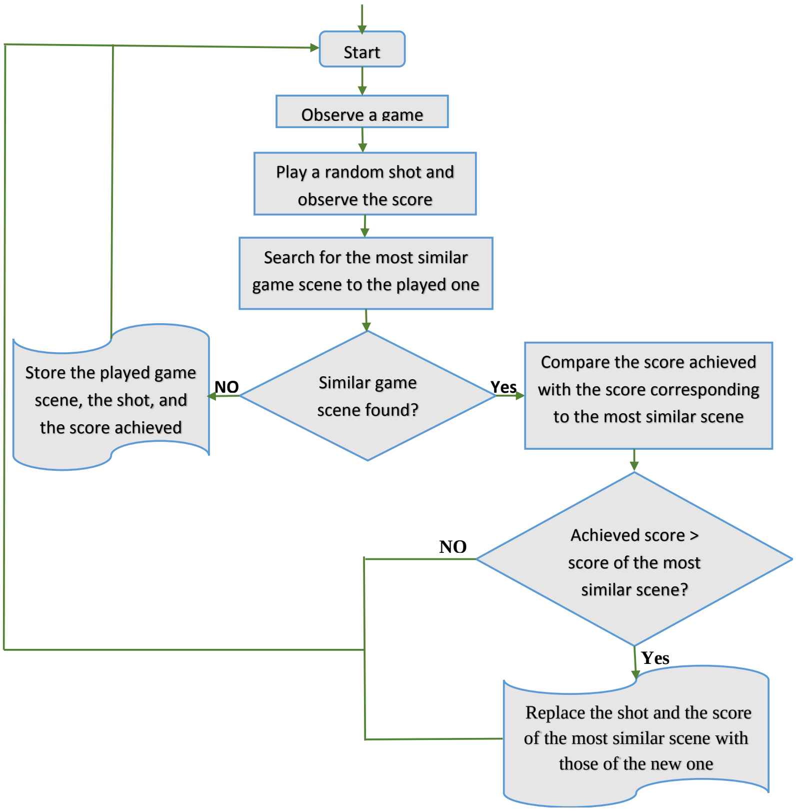

Fig. 1. The steps of the case base construction process.

For describing the rectangles, we adopt the interval-based representation, where a rectangle in the 2-dimensional space $\mathbb{R}^2$ has the following form: $R = [l, u] = [l_1, u_1] \times [l_2, u_2]$, where $l = (l_1, l_2)$ and $u = (u_1, u_2)$ are the coordinates of the lower left and upper right vertex of R, respectively. A complete game scene is represented through the set of the MBRs of all objects, together with their type when an object and colour when a bird.

Besides the game scene, collecting the cases also involves recording shots, which constitute the solution part of a case. In the Angry Birds Basic Game Playing Software, a shot is represented in the form of a 6-dimensional vector $s = (x, y, d_x, d_y, t_{shot}, t_{tap})$, where (x, y) and $(x + d_x, y + d_y)$ are the coordinates of the focus and release point, respectively, t_{shot} specifies the releasing and t_{tap} the tapping time of the bird in milliseconds.

To illustrate how a case is constructed, we consider the situation shown in Figure 2. The start game scene is shown in the picture on the left. The resulting scene after performing the shot with the trajectory indicated by the red line is shown in the picture on the right, where the change in score is seen as well.

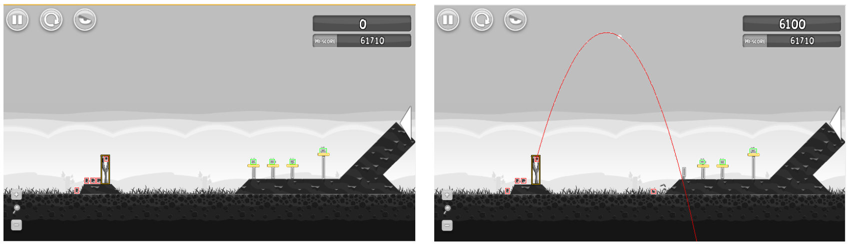

Fig. 2. The game scene before (left) and after (right) performing the shot indicated by the red line in the figure on the right. The MBRs of all objects in both scenes are marked. The change in score after performing the shot is shown on the top right of the figure on the right.

The case extracted from this scenario will contain the original scene, the performed shot and the achieved score, which we represent as follows:

`Sling:`	$l_1 = 200,\ u_1 = 305,\ l_2 = 216,\ u_2 = 363.$
`BirdType:`	*RedBird*.
`Hills:`	
`Hill 1:`	$l_1 = 471,\ u_1 = 237,\ l_2 = 839,\ u_2 = 384.$
`Pigs:`	
`Pig 1:`	$l_1 = 645,\ u_1 = 290,\ l_2 = 659,\ u_2 = 300.$
`Pig 2:`	$l_1 = 504,\ u_1 = 314,\ l_2 = 514,\ u_2 = 321.$
`Pig 3:`	$l_1 = 543,\ u_1 = 313,\ l_2 = 353,\ u_2 = 323.$
`Pig 4:`	$l_1 = 584,\ u_1 = 313,\ l_2 = 595,\ u_2 = 323.$
`TNTs:`	-
`Blocks:`	
`Block 1:`	$l_1 = 651,\ u_1 = 309,\ l_2 = 654,\ u_2 = 352.$
`Block 2:`	$l_1 = 509,\ u_1 = 330,\ l_2 = 513,\ u_2 = 351.$
`Block 3:`	$l_1 = 548,\ u_1 = 330,\ l_2 = 552,\ u_2 = 351.$
`Block 4:`	$l_1 = 588,\ u_1 = 330,\ l_2 = 591,\ u_2 = 350.$
`Block 5:`	$l_1 = 643,\ u_1 = 302,\ l_2 = 663,\ u_2 = 304.$
`Block 6:`	$l_1 = 500,\ u_1 = 325,\ l_2 = 520,\ u_2 = 327.$
`Block 7:`	$l_1 = 540,\ u_1 = 325,\ l_2 = 560,\ u_2 = 327.$
`Block 8:`	$l_1 = 579,\ u_1 = 325,\ l_2 = 599,\ u_2 = 327.$
`Shot:`	$x = 208,\quad y = 315,\ d_x = 35,\ d_y = 868,\ t_{shot} = 0,\ t_{tap} = 0.$
`Score:`	6100.

3.3 Case Retrieval

When the agent is playing, it gets a representation of the current game scene, searches the case base for the case with the most similar game scene and adopts its shot. Therefore, an appropriate measure to assess the similarity respectively dissimilarity between two game scenes is a key prerequisite for a successful agent. We compute the overall dissimilarity between two game scenes as the sum of the dissimilarities between their individual components:

$$\begin{aligned}
diss(scene_1, scene_2) = \; & diss(scene_1.Sling, scene_2.Sling) \\
& + diss(scene_1.BirdType, scene_2.BirdType) \\
& + diss(scene_1.Hills, scene_2.Hills \\
& + diss(scene_1.Pigs, scene_2.Pigs) \\
& + diss(scene_1.TNTs, scene_2.TNTs) \\
& + diss(scene_1.Blocks^S, scene_2.Blocks^S) \\
& + diss(scene_1.Blocks^W, scene_2.Blocks^W) \\
& + diss(scene_1.Blocks^I, scene_2.Blocks^I) \;\; ,
\end{aligned}$$

where $Blocks^S, Blocks^W$ and $Blocks^I$ denote blocks of stone, wood and ice, respectively.

The dissimilarity of two slings is just the dissimilarity between their MBRs. For the bird type, we compute the dissimilarity as follows:

$$diss(scene_1.BirdType, scene_2.BirdType) = \begin{cases} 0, & \text{if the types are equal,} \\ constant, & \text{otherwise.} \end{cases}$$

Measuring the dissimilarity between two game scenes in each of the remaining components (hills, pigs, TNTs, and blocks) reduces to measuring the dissimilarity between the two sets of rectangles, with potentially different cardinality, corresponding to the MBRs surrounding them. This requires building pairs from the elements of the two sets, between which the dissimilarity is to be computed. The overall dissimilarity between the two sets is then the sum of the dissimilarities between all pairs. We formulate the task of computing the dissimilarity between two sets of rectangles as a (potentially unbalanced) linear assignment problem, where the agents are the elements of one set, tasks are the elements of the other set and the total cost of an assignment is the overall sum of the dissimilarities between all built pairs.

In the following, we proceed with the description of the measure we use for assessing the dissimilarity between two rectangles, prior to detailing our approach to computing the dissimilarity between two game scenes in the above-mentioned components through solving appropriate assignment problems.

Dissimilarity Between Two Rectangles. Different measures exists to assess the dissimilarity between two rectangles in a p-dimensional space. We use the vertex-type distance d_v [2], which is defined for two 2-dimensional rectangles $R_1 = \left[l^{(1)}, u^{(1)}\right] = \left[l_1^{(1)}, u_1^{(1)}\right] \times \left[l_2^{(1)}, u_2^{(1)}\right]$ and $R_2 = \left[l^{(2)}, u^{(2)}\right] = \left[l_1^{(2)}, u_1^{(2)}\right] \times \left[l_2^{(2)}, u_2^{(2)}\right]$, as follows:

$$d_v\left(R_1, R_2\right) = \left(l_1^{(1)} - l_1^{(2)}\right)^2 + \left(u_1^{(1)} - u_1^{(2)}\right)^2 + \left(l_2^{(1)} - l_2^{(2)}\right)^2 + \left(u_2^{(1)} - u_2^{(2)}\right)^2 \;\; .$$

Dissimilarity Between Two Sets of Rectangles. As stated above, we build on solving an assignment problem to compute the dissimilarity between two sets of rectangles, which represent the MBRs of objects of specific material in two game scenes to be compared.

The linear assignment problem consists of mutually assigning objects of two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ in a cost-optimal manner. Formally, assignment costs are defined in terms of a matrix $C = (c_{ij})$, where c_{ij} denotes the cost of assigning a_i to b_j (and vice versa), $i.j \in [N] = \{1, \ldots, N\}$. The goal, then, is to find an assignment that minimizes the total cost

$$\sum_{i \in [N]} \sum_{j \in [N]} c_{ij} x_{ij}$$

with

$$x_{ij} = \begin{cases} 1, & \text{if } a_i \text{ and } b_j \text{ are mutually assigned,} \\ 0, & \text{otherwise.} \end{cases} \quad ,$$

subject to the following constraints:

$$\sum_{j \in [N]} x_{ij} = 1 \text{ for all } i \in [N],$$

$$\sum_{i \in [N]} x_{ij} = 1 \text{ for all } j \in [N],$$

The Hungarian algorithm [6] is one of the best-known methods for solving the assignment problem. It is mainly based on the observation that adding or subtracting a constant from all the entries of a row or a column of the cost matrix does not change the optimal solution of the underlying assignment problem. Thus, the algorithm proceeds iteratively, subtracting and adding constants in each step to specific rows and columns of the cost matrix, in such a way that more and more zero-cost pairs are built, until an optimal solution can be found. We refer to [6] for a detailed description of the Hungarian algorithm.

In the simplest form of the assignment problem, the number of objects in A and B are equal. For the problem at hand, this assumption does not hold; instead, we are dealing with an unbalanced assignment problem. To handle such problems, one usually introduces dummy rows or columns in the cost matrix, depending on which number exceeds the other. Normally, the introduced entries are filled with zeros, but this does not fit our purpose, because the addition or removal of objects will normally influence the best shot in a scene. We overcome this issue by associating a penalty with objects that remain unassigned. The penalty term for an unassigned rectangle is its distance to the zero-perimeter rectangle located at the origin, i.e., $R = [0, 0] \times [0, 0]$.

4 Experimental Results

We begin our experimental analysis with the construction of the case base, in which we proceed as follows. We run a random agent that chooses the coordinates of the shot to be executed fully at random, and we restrict ourself to the first 21 levels of the "Poached Eggs" episode of Angry Birds. The agent plays each level several times and the cases from each level are first collected in separate files. The distribution of the number of cases we gathered over the different levels of the game, shown in Table 1, was not uniform. That is, we dedicate more examples to harder levels than to easier ones. At the end, we combine all cases in one file, ending up with a case base of total size of $11,703$, which serves as the main case base for our agent.

Table 1. The number of cases we collected in each of the 21 levels of the game.

Level	# cases
1	50
2	50
3	50
4	130
5	50
6	100
7	50

Level	# cases
8	50
9	100
10	647
11	50
12	50
13	182
14	100

Level	# cases
15	50
16	50
17	50
18	400
19	200
20	100
21	100

After the case base was constructed, we first tested the performance of our agent on the above-mentioned levels. To this end, we let the agent play 10 games and report the minimal, maximal, and average score for each level, together with the standard deviation, in Table 2.

To get an idea of how our agent performs in comparison to others, Figure 3 plots the average score of our agent from Table 2 together with the scores of the naive agent, the top-3 agents of the 2013 and 2014 participants of the AI competition, and the average scores of all 30 participants, on all 21 levels, based on the 2014 benchmarks provided on the *aibirds.org* website. This comparison shows that our agent clearly outperforms both the naive and the average agent in both per-level and total scores, and is even competitive to the top-3 agents.

5 Conclusion and Future Work

We made use of CBR to build an Angry Birds playing agent. The results of an experimental study, in which we compared our agent with others, including the top-3 systems of previous AI competitions, are very promising, especially in light of the rather simple implementation of our agent so far. In fact, we are convinced that our agent's performance can be further enhanced through the collection of more cases and the refinement of the different steps of the CBR cycle.

More concretely, this work can be extended along the following directions. First, the real shape instead of the MBR representation can be used to represent the objects involved in the game. Second, a weighted version of the distance measure between game scenes can be learnt. Third, cases from levels of the

Table 2. The minimal, maximal, and average score, and the standard deviation of our agent in 10 games on the first 21 levels of the "Poached Eggs" episode of Angry Birds.

Level	Min. score	Max. score	Mean score	Standard deviation
1	28950	30790	29735	704.955
2	60950	61520	61293	188.388
3	42510	42540	42529	11.005
4	10660	36810	22500	9174.102
5	59680	67760	65301	2302.744
6	18020	35620	32096	6115.800
7	31180	46200	42486	5777.303
8	54110	54120	54111	3.162
9	32130	50020	44525	5874.565
10	32650	59920	46980	9294.536
11	54130	57390	55634	910.668
12	53010	54880	54248	550.713
13	21530	48090	33036	8987.933
14	49250	73760	65553	6858.706
15	37760	48540	46486	3166.492
16	54410	64790	61646	3073.714
17	46290	49900	48492	1224.444
18	39710	60830	49888	7150.137
19	31710	38550	33127	1999.445
20	34030	59140	46527	10113.806
21	59720	96880	70332	11020.633

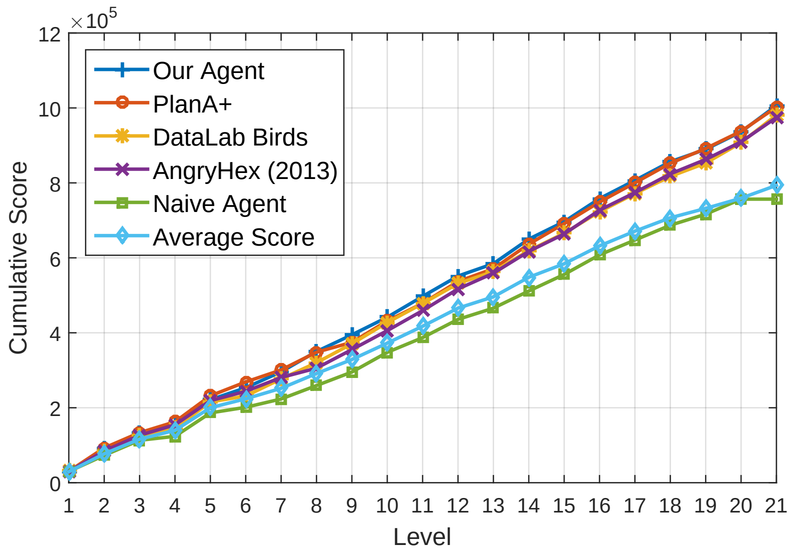

Fig. 3. The cumulative scores of our agent, the top 3 agents of the 2013 and 2014 participants of the AI competition, the naive agent, and the average agent, on the first 21 levels of the "Poached Eggs" episode of Angry Birds.

game other than the ones of the "Poached Eggs" episode can be extracted to increase the size and coverage of the case base. Fourth, since our agent does not realize any adaptation of the retrieved solutions so far, a sophisticated adaptation strategy could be another means to improve performance.

Acknowledgments. This work has been supported by the German Research Foundation (DFG).

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. *Artificial Intelligence Communications* 7(1), 3959 (1994)
2. Bock, H.H. Analysis of symbolic data: Exploratory methods for extracting statistical information from complex data. E. Diday (Ed.). *Springer-Verlag New York*, Inc., Secaucus, NJ, USA (2000)
3. Ferreira, L.A., Lopes, G.A.W., Santos, P.E.: Combining qualitative spatial representation utility function and decision making under uncertainty on the Angry Birds domain. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
4. Ge X., Gould, S., Renz, J., Abeyasinghe, S., Keys, J., Wang, A., Zhang, P. Angry Birds basic game playing software, version 1.32. Technical report. Research School of Computer Science, The Australian National University (2014)
5. Hüllermeier, E., Schlegel, P.: Preference-based CBR: First steps toward a methodological framework. In: Ram, A., Wiratunga, N. (eds.) *ICCBR 2011*. LNCS, vol. 6880, pp. 7791. Springer, Heidelberg (2011)
6. Kuhn, H. W. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2: 8397 (1955)
7. Lin, S., Zhang, Q., Zhang, H.: Object representation in Angry Birds game. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
8. Narayan-Chen, A., Xu, L., Shavlik, J. An empirical evaluation of machine learning approaches for Angry Birds. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
9. Polceanu, M., Buche, C.: Towards a theory-of-mind-inspired generic decision-making framework. In: IJCAI 2013 Symposium on AI in Angry Birds (2013)
10. Tziortziotis, N., Papagiannis, G., Blekas, K. A Bayesian ensemble regression framework on the Angry Birds game. In: *ECAI 2014 Symposium on Artificial Intelligence in Angry Birds* (2014)
11. Wałęga, P., Lechowski, T., Zawidzki, M. Qualitative physics in Angry Birds: first results. In: *ECAI 2014 Symposium on Artificial Intelligence in Angry Birds* (2014)
12. Zhang, P., Renz, J. Qualitative spatial representation and reasoning in Angry Birds: first results. In: *IJCAI 2013 Symposium on AI in Angry Birds* (2013)
13. Zhang, P., Renz, J. Qualitative spatial representation and reasoning in Angry Birds: the extended rectangle algebra. In: *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning*, KR14, p. to appear, Vienna, Austria (2014)