A Bayesian Ensemble Regression Framework on the Angry Birds Game

Nikolaos Tziortziotis, Georgios Papagiannis and Konstantinos Blekas
Department of Computer Science & Engineering, University of Ioannina, Greece
email: {ntziorzi,gpapagia,kblekas}@cs.uoi.gr

Abstract

In this article we introduce AngryBER, an intelligent agent architecture on the Angry Birds domain that employs a Bayesian ensemble inference mechanism to promote decision making abilities. It is based on an efficient tree-like structure for encoding and representing game screenshots, where it exploits its enhanced modeling capabilities. This has the advantage to establish an informative feature space and translate the task of game playing into a regression analysis problem. A Bayesian ensemble regression framework is presented by considering that every combination of objects' material and bird type has its own regression model. We address the problem of action selection as a multi-armed bandit problem, where the Upper Confidence Bound (UCB) strategy has been used. An efficient online learning procedure has been also developed for training the regression models. We have evaluated the proposed methodology on several game levels, and compared its performance with published results of all agents that participated in the 2013 and 2014 Angry Birds AI competitions. The superiority of the new method is readily deduced by inspecting the reported results.

Index Terms

Angry Birds game, Tree-like structure representation, Bayesian linear regression, Multi-armed bandit problem

I. Introduction

Physics-based simulation games such as Angry Birds, have received considerable and increasing attention during the last years. They are based on a simulator that has complete knowledge about the physical properties of all objects of the game world. This makes these games quite realistic, as they are able to simulate each move and its consequences to the real world with high precision. Despite the fact that these games are seemingly simple at a first glance, that is far from true. The extremely large or infinite number of the available actions, makes the particular games demanding and simultaneously attractive. The large number of moves stems from the fact that small deviations may result in differences in the outcome of the physics simulation. At the same time, it is really hard to predict the actions outcome in advance without an explicit knowledge of the games physical properties. In addition, it becomes even harder in the case where the game scenes can only be observed through a vision system, which corresponds to how humans are perceiving these games. Consequently, it becomes clear that a number of issues have arisen which demand new techniques that can investigate the fundamental physical game processes, so as to establish efficient AI agents which will be able to play as good or better than the best human players.

Angry birds was first launched in 2009 by Rovio(TM), and since then it has become one of the most popular games. The objective is to shoot birds using a slingshot in a way that all pigs are killed with the fewest number of used birds. Pigs are usually protected by complicated structures consisting of various types of building materials that must be destroyed. Several types of birds are available, with some of them being more effective against particular materials. At each time step, the point where the bird is released from the slingshot must be selected. In addition, the player has to decide about the exact tap time during the flight of the bird where its optional special feature will be activated rendering the bird more effective. The score (or return) achieved after each shot is calculated in terms of the number of the killed pigs and unused birds, as well as the extent of the destruction of each structure. The fewer birds are used as well as the more damage to the structures achieved, the higher the received score (or return).

Due to its nature (e.g. large state and action spaces, continuous tap timing, various objects' properties, noisy object detection, unpredictable action effects, etc.), Angry Birds constitutes a really challenging task for the development of intelligent agents. At the same time, the Angry birds competition[1] (AIBIRDS [1]) provides a very attractive venue where various AI agents compete with each other and evaluate their performance by playing in unknown game levels. A basic game platform [2] based on the Chrome version of the Angry Birds is provided by the organisers, incorporating a number of available components such as computer vision, trajectory planning and game playing interface. It should be stressed that the aforementioned platform has also been used for the purpose of developing our proposed agent.

[1]https://aibirds.org/

A. Related work

During the last two years, a number of interesting approaches have been proposed which are focused on the development of AI agents with playing capabilities similar to those exhibited by expert human players. These works rely on various AI techniques, such as logic programming, qualitative reasoning, advanced simulation, structural analysis, analysis of predicted damage, and machine learning methodologies.

In [3], [4], [5], the qualitative spatial representation and reasoning framework of [6] has been adopted for extracting relationships among scene objects. In [3], an extension of Rectangle Algebra [7] has been proposed for the determination of structure properties, such as its stability or the consequences after some external influences act. On the other hand, in [4] a qualitative physics method has been presented for the examination of the structure properties. In these two works, the action selection was made by measuring all possible shots in terms of a heuristic value function which depends on the shot's influence on the structures. On the other hand, in [5] a decision making under uncertainty scheme was applied for selecting of the most appropriate target according to an utility function. The main advantage of the aforementioned works is their ability to analyse the building blocks that consists the structure of a game scene, according to a number of factors, such as stability, destruction impact, connection points etc.. Therefore, they are able to discover the weaknesses of a building block as well as the destruction induced on the structure due to its demolishment.

An alternative work has been presented in [8] that employs the Weight Majority algorithm and the Naive Bayesian Network for selecting the most appropriate shot at each time step. However, a disadvantage on this scheme is that the constructed feature space is extremely large, since it incorporates a large amount of information about the game scene. In addition, it requires a huge amount of training data to be gathered in advance by using a number of different playing agents. Also, an extra effort is needed so as to manually label input data as positive (shots in winning games) and negative (shots in losing games) examples. Another work has been described in [9] based on a model-based methodology for learning the physical model of the world. For this reason, a number of trajectories are evaluated in the approximated model by performing a maximum impact selection mechanism. Its main characteristic is its ability to consider a large number of different trajectories and to the most appropriate one.

Finally, two quite similar agents that won the 2013 and 2014 AIBIRDS competitions are presented in [10] and [11], respectively. The specific agents combine a number of different strategies, some of which are simple, and select the most appropriate one according to the structure of the game scene. In this way, as each level is quite different, it has been shown that trying different strategies is much more possible to discover the strategy that suits quite well to a specific level.

B. Proposed Scheme

In this work, we propose a Bayesian ensemble regression framework for designing an intelligent agent for the Angry Bird domain. The novelty of the proposed methodology lies in the construction of an informative encoding scheme of the game scenes, as well as its ability to make accurate predictions and measure the effectiveness of each possible target through a compact ensemble model. These aspects are very important since they manage to build a low complexity agent, rendering it applicable in a real-time game such as Angry Birds.

Our methodology consists of the following two building blocks:

- Firstly, a novel tree-like structure is proposed for mapping scenes of game levels, where the nodes represent different material of solid objects. More specifically, each node of the tree depicts solitary or merged adjacent objects, which are constructed by the same material. This scene representation is informative as incorporates all the necessary knowledge about game snapshots, and simultaneously abstract so as to reduce the computational cost accelerating the learning procedure. The specific tree-like representation allows the construction of an efficient and simultaneously powerful feature space that can be used next during the prediction process.
- Secondly, an ensemble learning approach [12] is designed where every possible pair of 'object material' - 'bird type' has its own Bayesian linear regression model for the estimation of the expected return. In this way, the prediction ability of our scheme becomes much more accurate as it is able to distinguish possible associations between different types of birds and objects' materials. An ensemble integration framework based on the UCB algorithm [13] has been employed, using the predictions of every regressor to obtain the final ensemble prediction. After each shot, an online learning procedure is executed in order to adjust the model parameters of the selected regressor.

As experiments indicate, the proposed agent offers both flexibility and robustness, achieving superior modeling solutions. Additionally, our agent is compared with the results of all teams participated in the last two AIBIRDS competitions, as well as with the naive agent, which is a sufficient baseline evaluation criterion. In all cases, the provided experimental results prove the superiority of our solutions.

The remainder of this paper is organised as follows. The general framework of our methodology is described step-by-step in Sections II and III. Section II presents the proposed tree-like structure, the feature extraction and the *feasibility* property of the tree nodes. Furthermore, Section III describes the decision making mechanism, which is composed by a Bayesian ensemble scheme of linear regression models. In addition, the learning process for updating the model parameters is also discussed. In Section IV, we assess the performance of the proposed methodology reporting results obtained by applying our method to levels of the 'Poached Eggs' game set. Finally, in Section V we summarize the conclusions of this paper and give suggestions for future work.

II. Knowledge Representation

The proposed methodology is focused on describing the game scenes with an appropriate and useful structure so as to build an efficient state space representation. In addition, a decision making mechanism has been designed using a Bayesian ensemble regression framework that offers robustness and adaptability to dynamically changing situations. This is quite important as the levels in the Angry Birds are completely different to each other, while each one of the shots produces extremely different game scenes. Our work is based on the Angry Bird Game Playing software (version 1.32) [2].

Figure 1 illustrates briefly the main building blocks of the proposed approach. The whole procedure is repeated after each shot, every time a bird is available at the slingshot. In the following, a step-by-step description of the proposed agent architecture is presented:

1) Construct the **tree-like structure** of the game scene and establish a feature space.
2) Examine the **nodes feasibility** in terms of their ability to be reached (possible targets).
3) Predict the expected return of each feasible node (possible target) according to a **Bayesian Ensemble Regression** scheme, which takes into account the type of the object's material that corresponds to the node and the bird available on the slingshot. The most appropriate node is then selected as target.
4) **Tap timing** selection and perform shooting.
5) Adjust the model parameters of the selected regressor using an online **learning procedure**.

In the rest of this section as well as in the next one, we meticulously describe the specific parts of our methodology.

A. A tree-like structure representation of game scenes

The computer vision module of the AIBIRDS competition platform is used to analyse the video game scenes. It provides a list of all objects, $\mathcal{O}$, in the scene and identifies information about their type, location, and bounding box. The particular vision component can recognize the next seven (7) types of objects' materials:

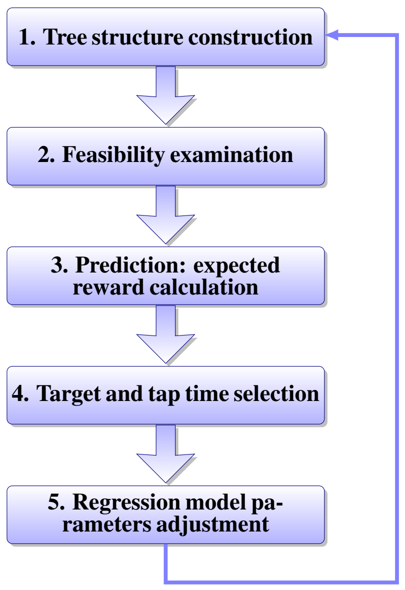
1. Tree structure construction
2. Feasibility examination
3. Prediction: expected reward calculation
4. Target and tap time selection
5. Regression model parameters adjustment

Fig. 1. Flow diagram of the proposed method.

Ice/Glass (**I**)	Wood (**W**)
Stone (**S**)	Rolling Stone (**RS**)
Rolling Wood (**RW**)	Pig (**P**)
TNT (**T**)	

The state space representation of the proposed method is based on the construction of an efficient tree structure in an attempt to arrange and manipulate all the scene objects and their attributes into a compact structure. It consists of a number of nodes that represent different spatial objects of the scene and a number of edges between them that signify their relations.

Algorithm 1 sketches the main steps for the construction of the proposed tree structure representation. The proposed tree structure is created through a three-stage process. Firstly, the tree nodes that correspond to the objects of a game scene are created and are positioned at the corresponding level. A complete tree-like structure of the game scene is designed by scanning a snapshot in the horizontal direction starting from the ground (level 1). Each time a different object is encountered, a new node is added to an appropriate level of the tree. After the creation of a node for each object in a game scene, a *virtual* root node is created at the highest level above all the other nodes.

After building the initial tree-like structure, a tree reduction procedure is performed. During this phase, we traverse the tree and merge nodes of either adjacent or same levels, in a recursive manner. Merging is done between nodes that have the same material type, are (approximately) adjacent and whose their

Algorithm 1: The Tree Structure Construction Algorithm

```
Input     : A list of the structure objects, $\mathcal{O}$
Output    : The tree structure, $\mathcal{T}$
Initialize: $k = 1$; $\mathcal{T} = \emptyset$
begin
    1. Complete tree-like structure construction phase.
    while $\mathcal{O} \neq \emptyset$ do
        Discover the object with the lowest center to the ground, $o \in \mathcal{O}$;
        Draw a straight horizontal line which passes through its center;
        Find the separated objects intersected by the line, $\mathcal{O}_k$;
        Sort $\mathcal{O}_k$ according to their positions at x-axis;
        Insert $\mathcal{O}_k$ at level $\mathcal{T}_k$, where each object $o \in \mathcal{O}_k$ is added as a separated node;
        $\mathcal{O} = \mathcal{O} \setminus \mathcal{O}_k$;
        $k++$;
    2. Add virtual root node at the highest level $k$;
    3. Reduction phase: Merge nodes of the same or adjacent levels according to type and shape
       properties;
    4. Create edges between tree nodes;
    5. Extract features from all tree nodes;
    return $\mathcal{T}$;
```

(vertical or horizontal) sides are of equal length. More specifically, two nodes are considered to be approximately adjacent if the distance between their sides is equal or less than 5 pixels in the actual game scene. Obviously, the merging procedure is not allowed for the object's type of pigs and TNTs. As it is expected, the merging procedure is capable of significantly reducing the size of the tree and therefore the computational cost of the decision making process, as it produces less possible targets for shooting. An example of this phase is shown in Fig. 4 for the game scene of Fig. 3, where the number of nodes is reduced from 30 (initial phase) to 18. In this example, the complete tree nodes $s_{71}, s_{72}, s_{81}, s_{82}$ that belong to the same or adjacent levels of the complete (left) tree, are merged into a single node (s_{51})

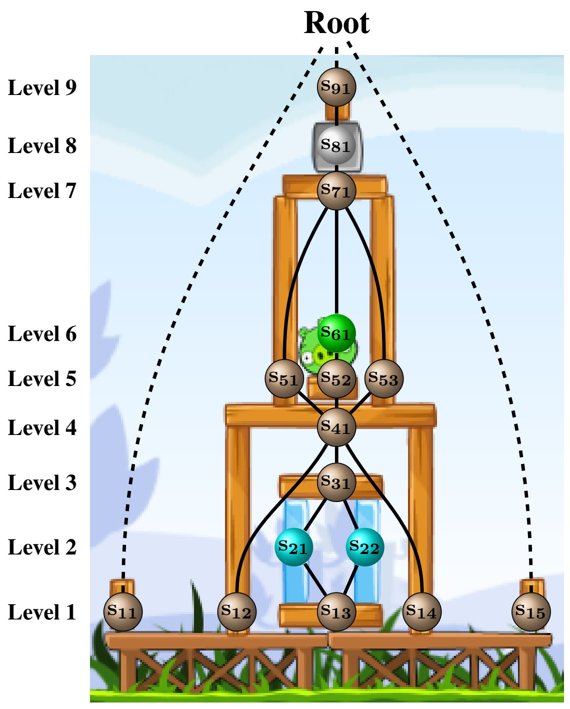

Fig. 2. The proposed tree-like structure consisting of 16 nodes at the first game level.

in the reduced (right) tree. Similarly, nodes s_{11}, s_{12} and s_{21} are merged together.

It must be noted that in the recursive procedure the direction (vertical or horizontal) for merging nodes does not play any significant role in most of cases, since it leads to the same final solution. However, in our experiments priority is given to the vertical direction, as we start merging nodes of adjacent levels. More specifically, nodes of adjacent levels (vertical direction) are merged first and in the subsequent step we merge nodes of the same level (horizontal direction). The whole procedure is repeated until no merging can be performed among tree nodes, see for example Fig. 3.

Finally, after the tree reduction phase, the complete tree of the game scene is designed. More specifically, edges among the tree nodes of different levels are created, declaring a relationship among them. In the tree-like structure, an edge between two nodes is created in the case where a segment of an object represented by a node lies above an object represented by a node of a lower level. At the same time, there must not be interfered an object represented by another node between them. In this way, a node is possible to have more than one parents, as objects of more than one nodes of higher levels can be located above the objects of that node. For example, in Fig. 2, node s_{41} has three parents as the objects represented by nodes s_{51}, s_{52} and s_{53} are located above it in the game scene. It is also worth noting that the nodes that correspond to roof objects, i.e. objects that do not have any other object above them; are

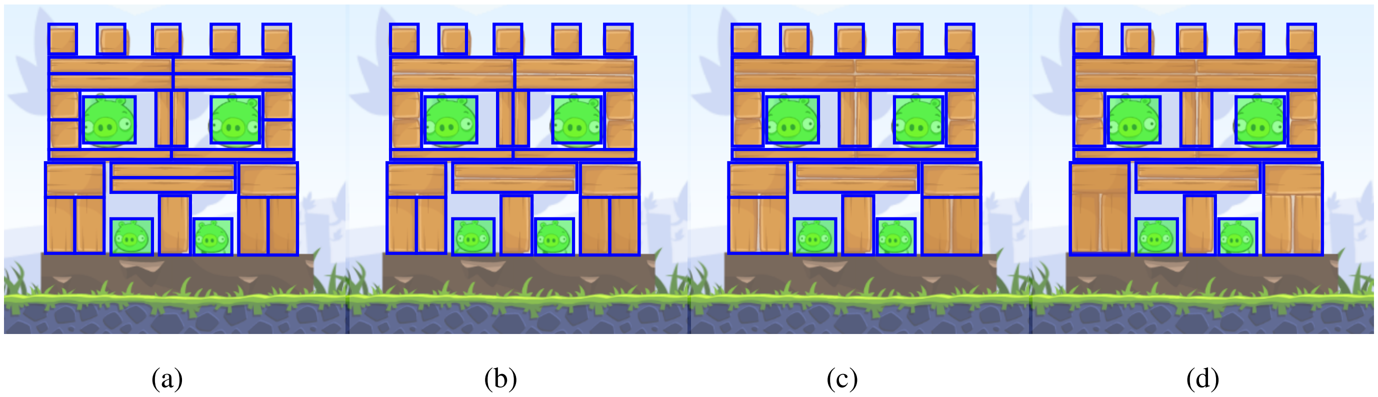

Fig. 3. A step-by-step representation of the tree *reduction* procedure at the 16th level of the 'Poached Eggs' season. Each bounding box illustrates an individual object or a group of adjacent objects of the same material, and corresponds to a tree node. (a) Tree nodes that corresponds to the initial tree-like structure. (b) At the second step, the tree nodes of adjacent levels (vertical direction) are merged. (c) At the third step, the tree nodes of the same level (horizontal direction) are merged. (d) Finally, the tree nodes of adjacent levels are concatenated if it is possible. The tree reduction phase is completed since no nodes are available for concatenation.

connected immediately with the root node. See for example nodes s_{11}, s_{15} and s_{91} in Fig. 2. Therefore, the proposed *tree-like* structure provides a convenient and attractive layout of the objects relationships, as well as a natural way for handling complex objects.

B. Feature Extraction

The tree-like structure framework allows us to extract quantitative features for each node s of the tree. These features can be used during the prediction process and are summarized as follows:

- $x_1(s)$: **Individual weight** calculated as the product of the object's area $Area(s)$ with coefficient c_s whose value depends on the material of the object, i.e. $x_1(s) = Area(s) \times c_s$. All types of objects have the same value for this coefficient, $c_s = 1$, except for Pig (P) and TNT (T) which have a much larger value ($c_s = 10$).
- $x_2(s)$: **Distance (in pixels) to the nearest pig**, normalized to $[0, 1]$ dividing the original distance by a threshold value for the maximum distance (100 in our case).
- $x_3(s)$: **Cumulative weight** calculated as the sum of individual weights of all ancestors $\mathcal{P}(s)$ of the node s in the tree, i.e. $x_3(s) = \sum_{s' \in \mathcal{P}(s)} x_1(s')$.
- $x_4(s)$: **Distance from the farthest ancestor**, normalized to $[0, 1]$ by dividing with a threshold value for the maximum height (e.g. 200).

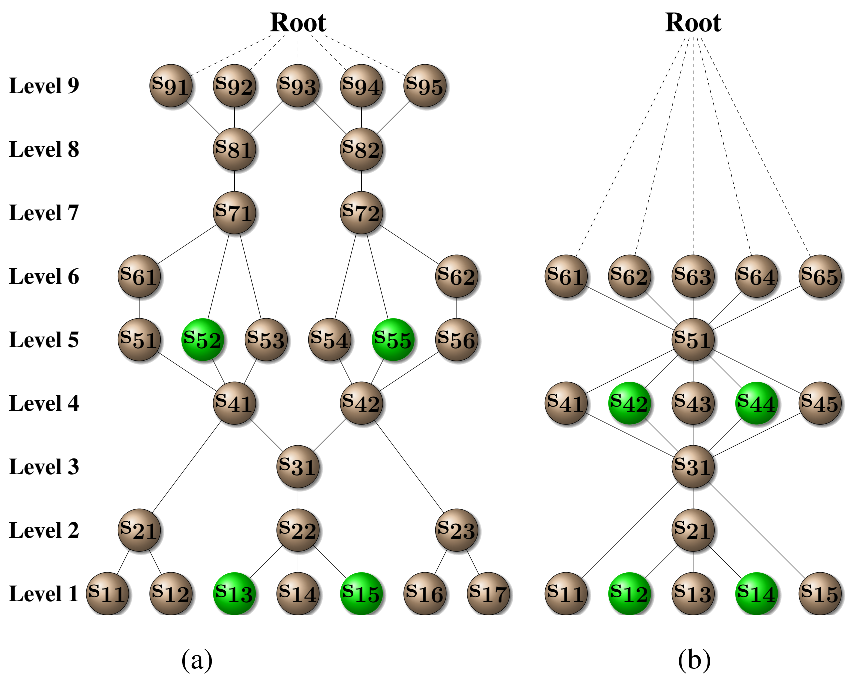

Fig. 4. Tree-like structure represention at the 16^{th} level of the ‘Poached Eggs’ season, before and after tree reduction phase. (a) The initial tree-like structure (Fig.3(a)). (b) The final tree-like structure (Fig.3(d)).

At this point, it is useful to present some technical information about the calculation of the above features. First of all, the objects weights used for the calculation of the individual weight, $x_1(s)$, have been selected empirically. A larger coefficient value ($c_s = 10$) is considered for both the Pig and TNT objects, in order to distinguish objects of these types with slightly different areas. For example, in Fig. 3, the areas of the pigs are nearly the same and thus, a clear distinction among them is not evident if a small coefficient value is used, such as $c_s = 1$. However, this is not the case with the other object materials, due to the merging procedure that is taking place during the construction of the tree-like structure. Moreover, the areas of the other object materials that encountered in a game scene presents much more diversity, in contrast to that of the Pig or TNT objects. Furthermore, it should be noted that the euclidean distances among the objects are calculated based on their centers. Finally, different threshold values for the maximum height and the maximum distance from a pig have been considered. In this context, our empirical analysis has shown that the above mentioned threshold values are the most suitable ones.

The above feature extraction strategy constructs an abstract but powerful feature space for all possible targets of a game scene. The proposed features are mostly spatial and concern information about geometrical, directional and topological properties of all tree nodes. An example can be seen at Table I that represents the features values of all 16 nodes of the tree illustrated in Fig. 2.

TABLE I

THE FEATURE VECTORS ALONG WITH THE FEASIBLE AND TYPE LABELS FOR THE 16 TREE NODES OF FIG. 2.

Node	Features					
	Level	Type	$x_1(s)$	$x_2(s)$	$x_3(s)$	$x_4(s)$
			Personal Weight	Pig Distance	Cumulative Ancestors Weight	Farthest Ancestor Distance
s_{11}	1	**W**	78	0.818	0	0
s_{12}	1	**W**	318	0.501	2467	0.65
s_{13}	1	**W**	188	0.660	5532	0.64
s_{14}	1	**W**	318	0.501	2467	0.645
s_{15}	1	**W**	78	0.818	0	0
s_{21}	2	**I**	156	0.504	2623	0.61
s_{22}	2	**I**	130	0.504	2623	0.61
s_{31}	3	**W**	156	0.341	2467	0.48
s_{41}	4	**W**	371	0.151	2096	0.385
s_{51}	5	**W**	318	0.164	416	0.36
s_{52}	5	**W**	72	0.082	556	0.35
s_{53}	5	**W**	318	0.198	416	0.36
s_{61}	6	**P**	140	0.170	416	0.32
s_{71}	7	**W**	182	0.431	234	0.1
s_{81}	8	**S**	156	0.521	78	0.065
s_{91}	9	**W**	78	0.651	0	0

C. Feasibility examination

The next step to our approach is to examine each node of the reduced tree-like structure in terms of its possibility to be reached (possible target). Reachability depends on the location of the material objects and stable obstacles.

Infeasible situations can happen in cases where an object is protected by a sheltering structure, making it not directly reachable for a bird, see for example Fig. 5(b). Moreover, it is possible for some stable obstacles in the path such as hills, to block a target (see for example the direct shot at Fig. 5(a)). Therefore, an examination step is initially required at each node of the tree so as to ensure that it is reachable.

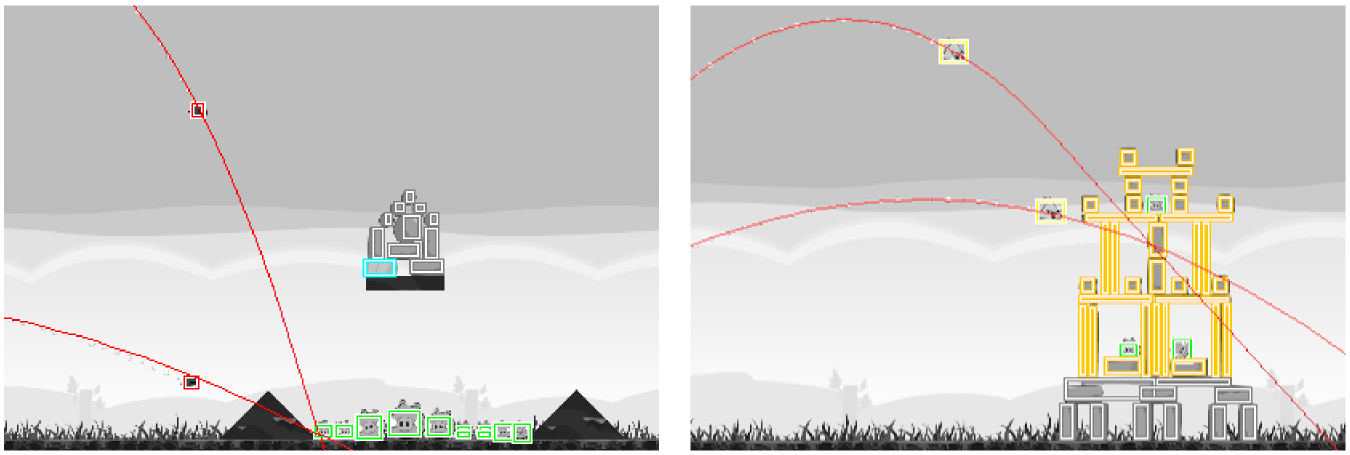
(a) (b)

Fig. 5. Tree's node feasibility examination. (a) Represents a feasible node (pig) as it is reachable by at least one trajectory. The direct shot is infeasible due to the fact that a hill is interposed between the slingshot and the target. (b) An infeasible node (wood) is represented as it is not directly reachable due to the tree-like structure.

Two different target points are considered for each tree node. The first one corresponds to the center of the left outer side and the second one to the center of the upper side of the node. Given a target point in the game scene, two different trajectories are returned by the trajectory planning module, which is provided by the AIBIRDS competition software platform: a) a direct shot (angle $<= 45°$) and b) a high arching shot (angle $> 45°$). Nevertheless, as the number of possible trajectories is infinite (infinite release points), a larger variety of trajectories can be also examined, which may lead to more effective shots[2]. In this way, for each one of the two possible target points of a tree node, we examine the above two mentioned trajectories in order to estimate the feasibility property of the node.

In the case where one of the possible targets (left or upper node side) of a node can be reached directly from at least one of the available two trajectories, the specific node is labelled as *feasible* (Fig. 5(a)). Otherwise, it is labelled as *infeasible* (Fig. 5(b)) and thus, it cannot be treated as a possible target. Roughly speaking, a shot is supposed to reach the target point directly, if its trajectory does not intersect any material object or stable obstacle located leftmost of the target object (see high arching shot in Fig. 5(a)). If both targets are reachable by at least one trajectory, priority is given to the target that corresponds to the longest side. Nevertheless, in the case where both sides -targets- of a node are of equal length, priority is given to the left side. Additionally, if both trajectories are accepted for the selected target,

[2]This was suggested by anonymous referees.

priority is given to the direct shot due to its effectiveness.

However, there are some special cases which are treated in a different way.

- In the case of the white bird a node is considered as feasible if its upper side can be reached directly by the bird's *egg* (Fig. 6), as opposed to the other types of birds.
- A tree node that represents a pig or a TNT is considered to be *feasible* even if the corresponding pig or TNT is protected by material objects, which form its sheltering structure. This is not true only in the case where a steady structure (e.g. hills) protects them. This is adopted as the pigs and TNTs are protected by other materials in most of the cases. In the opposite case, the agent will mainly focus on the destruction of the structures, without taking care of the pigs than remain alive and protected in a game scene.
- Finally, the nodes that correspond to objects which are located on the right of the rightmost pig, rolling stone or rolling wood, are characterised as *infeasible* irrespectively of being reachable or not.

After the feasibility examination phase, we end up with a set which contains only the tree's *feasible* nodes, denoted as $\mathcal{F}$. Only the nodes that belong at $\mathcal{F}$ ($s \in \mathcal{F}$) are considered as possible targets, thereafter. For example, after the feasibility examination process, the feasible nodes of the tree-like structure presented in Fig. 2 are $\mathcal{F} = \{s_{11}, s_{12}, s_{41}, s_{51}, s_{61}, s_{71}, s_{81}, s_{91}\}$.

III. Decision Making and Learning Process

The feasible nodes of the tree-like structure constitute a set of possible targets of the scene, i.e. points to be hit by a bird. In our approach the task of target selection has been made through an ensemble regression framework. Specifically, as shown previously, each feasible node $s \in \mathcal{F}$ is described with a feature vector $\boldsymbol{x}(s)$. We assume that during the game a sequence of game scores t resulting from each shot are observed. This can be seen as the target attribute that can be modeled using a linear regression scheme of the form:

$$t = \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}(s)) + \epsilon = \sum_{j=1}^{M} w_j \phi_j(\boldsymbol{x}(s)) + \epsilon. \tag{1}$$

In the above equation, M is the order of the regression model and $\boldsymbol{w} = (w_1, \dots, w_M)^\top$ is the vector of the M unknown regression coefficients. According to this equation, the score is represented as a linearly weighted sum of M fixed basis functions denoted as $\boldsymbol{\phi}(\boldsymbol{x}(s)) = (\phi_1(\boldsymbol{x}(s)), \phi_2(\boldsymbol{x}(s)), \dots, \phi_M(\boldsymbol{x})(s))^\top$. The error term, ϵ, is assumed to be zero mean Gaussian with variance $1/\beta$, i.e. $\epsilon \sim \mathcal{N}(0, \beta^{-1})$.

To construct the M basis functions we have considered the following strategy: First a number of samples (feature vectors) $\boldsymbol{x}(s)$ has been randomly collected from various game scenes. More specifically, a *random* agent has been used in a number of different levels, producing a variety of different game scenes. This agent selects a possible object material as target in a uniformly random way. Afterwards, various features vectors are extracted by using the game scenes, produced after the execution of a *random* shooting. Then, an hierarchical agglomerative clustering approach has been performed to the collected features creating an hierarchy of data clusters. In our scheme the standardized Euclidean distance was used as a criterion for merging pairs of clusters. At the end, a number of M clusters were selected from the agglomerative tree whose statistics were used for building the M basis functions and creating a kernel space. In our approach, we have considered normalized Gaussian kernels of the form:

$$\phi_j(\boldsymbol{x}(s)) = \exp\left(-\sum_{r=1}^{D}\frac{(x_r(s)-m_{jr})^2}{2\sigma_{jr}^2}\right) , \tag{2}$$

where D represents the size of feature space. Also, $\boldsymbol{m}_j = (m_{j1}, \ldots, m_{jD})$ and $\boldsymbol{\sigma}_j^{\mathbf{2}} = (\sigma_{j1}^2, \ldots, \sigma_{jD}^2)$ are the mean and variance of the j^{th} cluster, $\forall j = 1, \ldots, M$. It must be noted that the number of clusters M was not so crucial for the performance of the method. For our experiments, a different number of clusters M (e.g. $M \in \{75, 100, 125, \ldots, 250\}$) have been examined. Through our analysis it has been shown that $M = 150$ clusters are enough for the efficient representation of the state space. Moreover, we have noticed that having a reasonable number of clusters does not affect the online performance of our agent and keeps the complexity of the prediction process at reasonable levels.

Now consider a sequence of n input-target pairs of observations $\{(\boldsymbol{x}_1(s), t_1), \ldots, (\boldsymbol{x}_n(s), t_n)\}$. Given the set of regression model parameter values $\{\boldsymbol{w}, \beta\}$ we can model the conditional probability density of targets $\boldsymbol{t}_n = (t_1, \ldots, t_n)$ with the normal distribution, i.e.

$$p(\boldsymbol{t}_n|\boldsymbol{w}, \beta) = \mathcal{N}(\boldsymbol{t}_n|\Phi_n\boldsymbol{w}, \beta^{-1}I_n) , \tag{3}$$

where the matrix $\Phi_n = [\boldsymbol{\phi}(\boldsymbol{x}_1(s)), \boldsymbol{\phi}(\boldsymbol{x}_2(s)), \ldots, \boldsymbol{\phi}(\boldsymbol{x}_n(s))]^\top$ of size $n \times M$ is called design matrix and I_n is the identity matrix of order n.

An important issue when using a regression model is how to define its order M (number of basis functions). Models of small order may lead to underfitting, while large values of M may lead to overfitting. One approach to tackle this problem is through the Bayesian regularization framework [14], [15]. According to this scheme, a zero-mean (spherical) Gaussian prior distribution over weights w is considered:

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, a^{-1}I_M), \tag{4}$$

where the hyperparameter α is the inverse variance that controls the strength of the prior and I_M is the M-order identity matrix. Thus, the posterior distribution of the weights $\boldsymbol{w}$ is also Gaussian and can be obtained as:

$$p(\boldsymbol{w}|\boldsymbol{t}_n, \alpha, \beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_n, \Sigma_n) \ , \tag{5}$$

where

$$\boldsymbol{\mu}_n = \beta \Sigma_n \Phi_n^\top \boldsymbol{t}_n \quad \text{and} \quad \Sigma_n = (\beta \Phi_n^\top \Phi_n + aI_M)^{-1}, \tag{6}$$

are its mean value and the covariance matrix, respectively.

As mentioned previously, we are interested in making predictions for the score value. Suppose we have observed a sequence of n score values $\boldsymbol{t}_n = (t_1, t_2, \ldots, t_n)$. According to the regression model, when examining a feasible node $q \in \mathcal{F}$ of the tree (possible target) that has a feature vector $\boldsymbol{x}(q)$, we can obtain the posterior predictive distribution of its score t^q which is also Gaussian:

$$p(t^q|\boldsymbol{t}_n, \alpha, \beta) = \mathcal{N}(t^q|\boldsymbol{\mu}_n^\top \boldsymbol{\phi}(\boldsymbol{x}(q)), \sigma_{nq}^2) \ , \tag{7}$$

where

$$\sigma_{nq}^2 = \beta^{-1} + \boldsymbol{\phi}(\boldsymbol{x}(q))^\top \Sigma_n \boldsymbol{\phi}(\boldsymbol{x}(q)). \tag{8}$$

This prediction can be used to evaluate a possible target of a game scene by calculating the quantity:

$$\hat{t}^q = \boldsymbol{\mu}_n^\top \boldsymbol{\phi}(\boldsymbol{x}(q)) \ . \tag{9}$$

However, the decision about which is the optimum node to be selected as target depends on the material type of objects represented by a node, as well as the bird that is available to the slingshot. Hence, a separate regression estimator is used for every pair of material type-bird, so as to enhance the accuracy of the decision process. In our approach, we have applied an *ensemble* scheme of regressors, where every combination of material and bird type was assumed to have its own parametric regression model. Therefore, since there are 7 objects $\times$ 5 birds $= 35$ combinations, we have 35 different linear regression models with parameters $\theta_l = \{\boldsymbol{w}_l, \beta_l\}$, $l = 1, \ldots, 35$. Every time only a subset of these regressors become active based on the type of bird that is available on the slingshot and the type of object's material found in the game scene. The feasible nodes, $q \in \mathcal{F}$, of the tree are then evaluated by calculating the predicted reward value, $\hat{t}^q$, according to Eq. 9. This is achieved using the regression model $f(q)$ that corresponds to the pair of object material of node q and bird type.

The final step before generating a shot is to select the target among all feasible nodes, $q \in \mathcal{F}$, of the constructed tree. In our approach, we have approach the ensemble regression model as a multi-armed bandit model. The trade-off between the need to obtain new knowledge and to exploit the already obtained knowledge to improve performance is one of the most fundamental problems encountered in nature. In this direction, we have employed the Upper Confidence Bound (UCB) strategy [13] which offers a balance between exploration and exploitation dilemma during learning process. According to the UCB framework we maintain the number of times $n_{f(q)}$ that each arm (type of regressor, $f(q)$) has been played. The decision procedure is determined by maximizing the following function:

$$q^* = \arg\max_{q \in \mathcal{F}} \left\{ \left(\boldsymbol{\mu}_{n_{f(q)}}^{f(q)} \right)^\top \boldsymbol{\phi}(\boldsymbol{x}(q)) + C \sqrt{\frac{2 \ln N}{n_{f(q)}}} \right\} , \tag{10}$$

where N is the total number of plays so far (number of shots) and C is a tuning parameter of the UCB decision making process that is used to balance exploration and exploitation (during our experiments we have used $C = 3000$). Intuitively, the UCB framework manages to balance between selecting actions with good belief (targets with large reward prediction) and/or actions which have large uncertainty (small $n_{f(q)}$).

A. *Tap Timing*

After the selection of best among the tree's feasible nodes ($\mathcal{F}$), the tap timing procedure is executed. Using the trajectory planner component of the game playing framework the corresponding tap time is calculated in advance and a tapping is performed right before the estimated collision point. In our approach the tap time strategy depends on the type of birds used:

- Red birds (*Red*) is the leader of the flock, but do not have any special feature at their arsenal. Therefore, there is no need for tapping.
- Blue birds (*the Blues*) split into a set of three similar birds when the player taps the screen. The agent randomly performs a tap in an interval between 65% and 80% of the trajectory from the slingshot to the first collision object.
- Yellow birds (*Chuck*) accelerate upon tapping which is performed randomly between 90% and 95% of the trajectory in the case of high-arching shots (angle $> 45°$). In the case of direct shots (angle $<= 45°$), tap time is selected randomly between 85% and 90% of the trajectory.
- White birds (*Matilda*) drop *eggs* in the target below them. In this case, tapping is executed when the bird lies above the target (see, Fig. 6). As experiments have shown, this strategy is very efficient for handling the specific type of birds.
- Black birds (*Bombs*) are the most powerful member among the birds. No tapping is performed by the agent during the bird flight. The bird blow up in a short time period after impinging on a scene object.

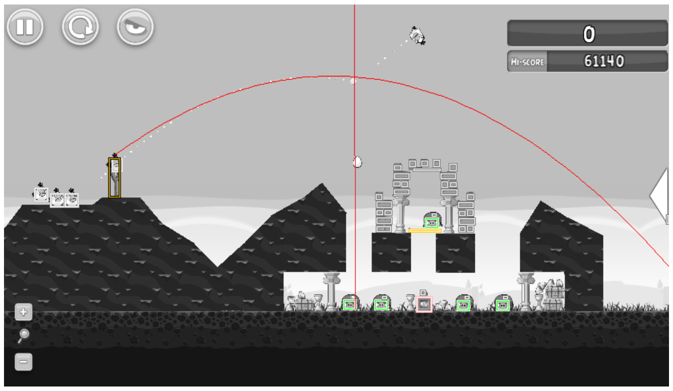

Fig. 6. Tap timing procedure for the white bird. Tapping is performed only when the bird lies above the target (pig).

B. Online learning of model parameters

The final step of the proposed scheme is the learning procedure. Due to the sequential nature of data, a recursive estimation framework has been followed for updating the regression model parameters [15]. This can be considered as an online learning solution to the Bayesian learning problem, where the information on the parameters is updated in an online manner using new pieces of information (rewards) as they arrive. The underlying idea is that at each measurement we treat the posterior distribution of previous time step as the prior for the current time step.

Suppose the tree node q^* has been selected based on the Bayesian ensemble mechanism (Eq. 10) that corresponds to the regression model $k \triangleq f(q^*)$. Then, the selection frequency, n_k, of this regressor is increased by one, we shoot the target and we receive a score, $t^k_{n_k+1}$. The last constitutes a new observation for the k^{th} regression model, i.e. $\boldsymbol{t}^k_{n_k+1} = (\boldsymbol{t}^k_{n_k}, t^k_{n_k+1})$ which is normally distributed:

$$p(t^k_{n_k+1}|\boldsymbol{w}^k) = \mathcal{N}(t^k_{n_k+1}|\boldsymbol{w}^{k\top}\boldsymbol{\phi}(\boldsymbol{x}(q^*)), \beta^k) \ , \tag{11}$$

where $\boldsymbol{x}_{n_k+1}(q^*)$ is the feature vector of the selected tree node, q^*.

We can now obtain the posterior distribution of weights $\boldsymbol{w}_k$, as:

$$\begin{aligned} p(\boldsymbol{w}^k|\boldsymbol{t}^k_{n_k+1}) &\propto p(t^k_{n_k+1}|\boldsymbol{w}^k)p(\boldsymbol{w}^k|\boldsymbol{t}^k_{n_k}) & (12)\\ &= \mathcal{N}(\boldsymbol{w}^k|\boldsymbol{\mu}^k_{n_k+1}, \Sigma^k_{n_k+1}) \ , & (13)\end{aligned}$$

where we can obtain the following recursive forms:

$$\Sigma^k_{n_k+1} = \left[(\Sigma^k_{n_k})^{-1} + \beta^k \boldsymbol{\phi}(\boldsymbol{x}(q^*))\boldsymbol{\phi}(\boldsymbol{x}(q^*))^\top\right]^{-1} \ , \tag{14}$$

$$\boldsymbol{\mu}^k_{n_k+1} = \Sigma^k_{n_k+1}\left[\beta^k \boldsymbol{\phi}(\boldsymbol{x}(q^*))\boldsymbol{t}^k_{n_k+1} + (\Sigma^k_{n_k})^{-1}\boldsymbol{\mu}^k_{n_k}\right] \ . \tag{15}$$

The above equations constitute an efficient recursive procedure for adjusting the model parameters of the winner regressor k, after shooting. That provides also the opportunity to monitor learning process. In the beginning of the game, (i.e. step 0) all the information we have about the parameters of all regression models, is the prior distribution $p(\boldsymbol{w}_k)$ which is assumed to be zero mean Gaussian ($\boldsymbol{\mu}_0^k = \mathbf{0}$) with spherical covariance matrix ($\Sigma_0^k = a^{-1}I_M$).

Algorithm 2 summarizes the basic steps of the proposed method for playing the Angry Bird game.

Algorithm 2: The **AngryBER** learning algorithm

```
while available bird on the slingshot do
    1. Game scene's objects detection;
    2. Tree-like structure construction of the game scene and feature extraction (Alg. 1);
    3. Tree nodes feasibility examination, $\mathcal{F}$ (Sec. II-C);
    4. Select the best target node among all the feasible nodes, $q^* \in \mathcal{F}$ according to Eq. 10. This
       corresponds to the regressor $k \triangleq f(q^*)$.
    5. Compute the most effective timing for the tapping execution (Sec. III-A);
    6. Hit the target and receive reward, $t_{n_k+1}$;
    7. Adjust the model parameters (Eqs. 14,15) of the selected regressor, $k$;
```

IV. Experimental Results

A series of experiments has been conducted in an effort to analyze the performance of the proposed agent (**AngryBER**) in the Angry birds domain. Due to the low complexity of the general framework where our agent is built up, the experiments have taken place in a conventional PC[3]. The source code of the agent can be found in [16]. Our analysis has concentrated mainly on the first 2 episodes from the freely available 'Poached Eggs' season of the Angry Birds game. Each one of the episodes consists of 21 levels, which have to be passed, in order to assume that the episode is successfully completed.

The following procedure was used for training the AngryBER agent. Ten (10) complete passes of the previously mentioned episodes have been sequentially executed. The agent remains at the same level if he

[3]Intel Core 2 Quad (2.66GHz) CPU with 4GiB RAM

fails to destroy all pigs found in the game scene. In order to evaluate our agent we have tried to comply with the AIBIRDS competition rules [17]. Therefore, the agent has at his disposal at least 3 minutes on average in order to complete a game level, corresponding to a total time of 63 minutes for each episode. It must be noted that the results have shown that our agent needs only a part of the available time for a successful episode completion.

In our experiments, we examine two variations of the AngryBER agent, named '**AngryBER$_\mathbf{1}$**' and '**AngryBER$_\mathbf{2}$**', respectively. The only difference between the two versions lies in the consideration of a diverse set of features (Sec. II-A) that are used to represent the feature space of each tree node. More specifically, both of them use the first two distinctive features $\{x_1, x_2\}$ that are referred to node's personal weight and its distance from the nearest pig, respectively. However, they differ in the way they examine the relation of nodes with their ancestors. Roughly speaking, the first variation '**AngryBER$_\mathbf{1}$**' takes into account the density of the structure that lies above each node (x_3), while the second '**AngryBER$_\mathbf{2}$**' considers the height of the structure located above object materials.

In order to compare the AngryBER agent, we have used the *naive* agent provided by [2], which uses an unsophisticated strategy. More particularly, the *naive* agent shoots the birds directly to the pigs, selecting randomly a pig as target, without any further reasoning. It has been shown that naive agent provides a sufficient baseline for agent's evaluation as it is indicated by the results provided at [18]. Moreover, we have compared our agent with the agents proposed by all teams participated in 2013 and 2014 AIBIRDS competitions [18] (30 teams in total). It is worth mentioning that for the $2nd$ episode only results of the last year teams (2014) are provided (10 teams in total), since an updated version of the vision system was released only last year. This vision system is able to detect the real shape of objects, ground and hills.

We have run 30 independent experiments for each variation of our agent. The results about the first two episodes of 'Poached Eggs' season are presented in Tables II and III, respectively. In these tables various measures of descriptive statistics about the score reached per level are provided, in an effort to obtain a more comprehensive comparison study. These are: the mean, median, minimum and maximum score found (measures of central tendency), as well as the standard deviation and the interquartile range $IQR = Q3 - Q1$, (measures of variability) of scores.

A number of interesting remarks stems from our empirical evaluation:

- The first one and most impressive observation is that both variants of our AngryBER agent succeed to pass every level with success. While it may seem easy at a first glance, it is far from true as lot of agents fail most levels, since the degree of difficulty increases continuously with every successfully completed level. For example, only 15 out of 30 agents (50%) achieve to complete the 21st level of the first episode. It becomes much more evident at the levels of the second episode. In this case, the agents achieve to pass approximately half of the levels (51% success rate). More specifically, *DataLab Birds* (best agent's performance) achieves to pass 17 out of 21 levels (80% success rate), while *S-birds Avengers* (worst agent's performance) achieves to complete only 3 levels (14% success rate).

- AngryBER obtains satisfactory scores in the majority of levels. According to the results, the proposed agent manages to reach (26) high scores at the levels of the two episodes: 7 and 19 high scores obtained at the levels of the first and second episode, respectively.
- Additionally, our agents achieve to gain '3-stars' in a considerably high percentage of visited levels. '3-stars' provides a baseline that indicates superior performance. Gaining '3-stars' could be considered as a measure of the agent's ability to destroy all pigs by using the least possible number of birds. More specifically, $\mathbf{AngryBER_1}$ and $\mathbf{AngryBER_2}$ have achieved to gain '3-stars' at the 71% and 81% of the levels, respectively. On the other side, both best-performed agents, DataLab Birds and Plan A+, have achieved to gain '3-stars' at 31% of the levels.
- Another interesting remark is that the mean scores of our agents are always better than those of the benchmarks, with a single exception for the first level of the first episode. Particularly, the expected total reward gained by $\mathbf{AngryBER_1}$ and $\mathbf{AngryBER_2}$ agents at the first episode is 914039 (11th place) and 967160 (3d place), respectively. On the other hand, the expected total reward gained by the proposed agents at the second episode is 1120459 (1st place) and 1154292 (1st place), respectively. Moreover, the best total reward that could be achieved by the $\mathbf{AngryBER_1}$ agent at the first and second episodes is 1055790 (1st place) and 1363840 (1st place), respectively. Additionally, the best total reward that could be achieved by the $\mathbf{AngryBER_2}$ agent at the two episodes is, 1096520 (1st place) and 1394856 (1st place), respectively. Another point that should be highlighted is that if we consider the median statistic, our agent performs significantly better than half of the agents provided by the benchmarks.
- Finally, robustness is a key feature of our method as indicated from the small values on both variability measurements (standard deviation and interquartile range) in most levels.

The presented experimental results highlight the superiority of both variations of our proposed approach over many existing methods. Nevertheless, the second variant, $\mathbf{AngryBER_2}$, performs slightly better than the first one, $\mathbf{AngryBER_1}$. This is more apparent in difficult levels. Moreover, $\mathbf{AngryBER_2}$ has reached

18 out of 26 high scores found by both agents, and has gained '3-stars' at 34 out of 42 levels in total, while $\mathbf{AngryBER_1}$ has gained '3-stars' at 30 out of 42 levels. Additionally, it becomes evident that the total return gained by $\mathbf{AngryBER_2}$ in both episodes, and especially in the case of the first one, is quite higher than that of $\mathbf{AngryBER_1}$. Thus, we conclude that the consideration of the height of the structure lying above a tree node (feature, x_4) seems to be more effective than the information of the structure's density (feature, x_3).

Another impressive characteristic of the proposed scheme is its ability to speed-up the learning process and discover good policies quickly. This is attributed to the efficient tree-like structure representation in combination with the ensemble learning strategy. In this context, AngryBER agent is robust and adaptive as it is able to identify the effectiveness of various bird types in destructing particular materials.

Finally, the '$\mathbf{AngryBER_1}$' agent has joined to the 2014 AIBIRDS competition managing to win the 2nd prize. Our competition participation, gave us the opportunity to assess the performance and generalization capability of our agent in unknown challenging levels. As it was proved, the AngryBER agent can cope with success in the most of the assigned levels. The competition results can been found in [19].

V. Conclusions and Future Work

In this work, we presented an advanced intelligent agent for playing the Angry Birds game, based on an ensemble of regression models. The key aspect of the proposed method lies on an efficient tree-like scene representation. This allows the exploitation of its superior modeling capabilities to establish a rich feature space. An ensemble scheme of Bayesian regression models is then proposed, where different regressors for each pair of bird-material type are combined and act in a competitive fashion. The target is then selected according to the UCB decision making process, which aims at gaining new knowledge by exploring its environment and exploiting its current, reliable knowledge. Learning procedure is achieved in terms of an online estimation framework. Experiments on several game levels demonstrated the ability of the proposed methodology to achieve improved performance and robustness compared to other approaches on the Angry Birds domain.

Although we have investigated the performance of the proposed method in a variety of challenging levels, we are planning to examine its generalization capabilities more systematically to more advanced levels. Since the tree-like structure is very effective and general, another future research direction is to examine the possibility of enriching the feature space with alternative topological features, which can be extracted from the proposed lattice structure, as the ones suggested by [3]. A general issue in the regression analysis is how to define the proper number of basis functions. Sparse Bayesian regression

TABLE II

PERFORMANCE STATISTICS AT THE 21 LEVELS OF THE FIRST ‘POACHED EGGS’ EPISODE

Level	**3 stars**	**Agent**	**Mean**	**Std**	**Median**	**IQR = Q3-Q1**	**Min**	**Max**
1	32000	**AngryBER$_1$**	29068	±116	29030	0	**29030**	29410
		AngryBER$_2$	28468	±73	28430	80	28430	28800
		Benchmarks	**29233**	±2682	**29635**	1360	18420	**32660**
2	60000	**AngryBER$_1$**	**51196**	±4009	52410	230	33930	53850
		AngryBER$_2$	47363	±6600	**52420**	9080	**34160**	52740
		Benchmarks	47932	±8844	52180	9620	26240	**62370**
3	41000	**AngryBER$_1$**	**41820**	±352	**41910**	0	**40260**	41910
		AngryBER$_2$	41804	±405	**41910**	0	**40260**	41910
		Benchmarks	39029	±4780	41280	1730	24070	**42240**
4	28000	**AngryBER$_1$**	21175	±3932	19190	1470	18720	29150
		AngryBER$_2$	**27471**	±3186	**29010**	1240	**18990**	**29010**
		Benchmarks	23458	±6675	21420	8690	10120	36810
5	64000	**AngryBER$_1$**	**63898**	±3028	**64460**	0	47870	64460
		AngryBER$_2$	63482	±422	63520	0	**61440**	64460
		Benchmarks	60508	±9078	64000	9340	35650	**70350**
6	35000	**AngryBER$_1$**	**34580**	±3222	**35640**	0	**24680**	35720
		AngryBER$_2$	33938	±4864	35385	60	15290	36470
		Benchmarks	23687	±10630	25850	17400	0	**36970**
7	45000	**AngryBER$_1$**	**33763**	±5115	**37000**	6600	**21760**	37140
		AngryBER$_2$	32933	±5874	36940	8690	20550	45780
		Benchmarks	28053	±14166	29350	14830	0	**49120**
8	50000	**AngryBER$_1$**	**47439**	±10407	**54730**	18310	26630	57130
		AngryBER$_2$	40835	±6644	38060	1830	**28710**	55630
		Benchmarks	39473	±13377	43260	20330	0	**57780**
9	50000	**AngryBER$_1$**	32227	±5758	31940	2900	23060	48780
		AngryBER$_2$	**43731**	±4083	**45220**	0	**32450**	48780
		Benchmarks	37300	±11914	38790	21710	0	**51480**
10	55000	**AngryBER$_1$**	46848	±7064	47810	9530	34240	59670
		AngryBER$_2$	**52939**	±8488	49720	13990	**34900**	68110
		Benchmarks	43805	±17857	**50910**	20080	0	**68740**
11	54000	**AngryBER$_1$**	44151	±1719	44860	1050	35610	44860
		AngryBER$_2$	**51347**	±2092	**51425**	1520	**43220**	53580
		Benchmarks	45052	±14644	48390	14780	0	**59070**
12	45000	**AngryBER$_1$**	**52352**	±3018	**53690**	1300	**39680**	54980
		AngryBER$_2$	47412	±7790	48985	15970	36990	56910
		Benchmarks	48348	±14306	53230	9540	0	**61070**
13	47000	**AngryBER$_1$**	26739	±5527	26580	8920	19450	39010
		AngryBER$_2$	**37725**	±7541	**37520**	11080	**21760**	49240
		Benchmarks	30125	±14480	31425	15150	0	**50360**
14	70000	**AngryBER$_1$**	60955	±6002	**65640**	10000	45640	**65640**
		AngryBER$_2$	**64177**	±3805	**65640**	0	**49370**	**65640**
		Benchmarks	52097	±18770	57805	12970	0	**65640**
15	41000	**AngryBER$_1$**	40549	±6401	39350	12360	31190	50020
		AngryBER$_2$	**44908**	±4756	**47370**	7050	**32480**	49620
		Benchmarks	36111	±16019	41390	13660	0	**55300**
16	64000	**AngryBER$_1$**	54554	±5612	52610	4390	45540	**70590**
		AngryBER$_2$	**65191**	±6622	**69590**	7570	**47540**	69590
		Benchmarks	47720	±22513	55600	11470	0	66570
17	53000	**AngryBER$_1$**	**47161**	±3391	**46850**	2050	**39090**	**55760**
		AngryBER$_2$	42631	±3360	42935	5510	37400	49900
		Benchmarks	39260	±16278	44970	8570	0	54750
18	48000	**AngryBER$_1$**	42668	±3961	43305	6810	36350	49790
		AngryBER$_2$	**51669**	±5297	**54180**	7380	**38860**	**56440**
		Benchmarks	35704	±18691	43435	10370	0	54500
19	35000	**AngryBER$_1$**	33681	±3763	35530	6750	**27570**	38090
		AngryBER$_2$	**35461**	±2817	**36690**	2910	25420	39220
		Benchmarks	25043	±15020	30425	20120	0	**40100**
20	50000	**AngryBER$_1$**	46094	±6784	45470	14500	36060	55590
		AngryBER$_2$	**50124**	±6517	**54040**	8250	**37170**	**58550**

TABLE III

Performance statistics at the 21 levels of the second ‘Poached Eggs’ episode

Level	3 stars	Agent	Mean	Std	Median	IQR = Q3-Q1	Min	Max
1	60000	**AngryBER$_1$**	51449	±5559	52635	9960	42770	61470
		AngryBER$_2$	**54363**	±5392	**56880**	7490	**43950**	**64820**
		Benchmarks	38922	±27610	48120	59990	0	64050
2	60000	**AngryBER$_1$**	52769	±2865	52795	3030	47140	59880
		AngryBER$_2$	**53203**	±2747	**53205**	3270	**47650**	58640
		Benchmarks	19510	±33934	0	45790	0	**96180**
3	102000	**AngryBER$_1$**	96909	±6007	98165	8070	86510	114160
		AngryBER$_2$	**100960**	±8532	**99240**	9400	**87730**	**116510**
		Benchmarks	69156	±47915	96375	99490	0	108510
4	50000	**AngryBER$_1$**	50784	±7366	**54540**	10420	26490	58910
		AngryBER$_2$	**51685**	±5862	52290	7680	**32230**	**59690**
		Benchmarks	26018	±27511	24290	52620	0	56550
5	80000	**AngryBER$_1$**	**84366**	±5676	**84775**	6410	**69730**	**93910**
		AngryBER$_2$	82075	±6506	83800	7720	66650	91320
		Benchmarks	43862	±37933	67440	74940	0	78810
6	62000	**AngryBER$_1$**	59301	±4998	57905	8230	51270	69500
		AngryBER$_2$	**60641**	±6010	**58765**	7470	**53990**	**72480**
		Benchmarks	34275	±24892	41640	56750	0	58270
7	50000	**AngryBER$_1$**	42047	±5583	43430	9710	**33260**	53740
		AngryBER$_2$	**48265**	±6839	**47980**	8960	31480	**62140**
		Benchmarks	38245	±20456	46855	8580	0	53450
8	53000	**AngryBER$_1$**	**49397**	±5494	47915	9550	**39470**	58380
		AngryBER$_2$	48942	±6170	**48325**	10840	38210	**59260**
		Benchmarks	34706	±24160	47340	50300	0	57300
9	28000	**AngryBER$_1$**	23239	±3288	22635	2370	19150	34430
		AngryBER$_2$	**24416**	±3274	**23340**	3620	**20420**	33310
		Benchmarks	21007	±13517	22650	6380	0	**46240**
10	40000	**AngryBER$_1$**	**39714**	±3674	**40720**	2280	**32940**	**49910**
		AngryBER$_2$	35459	±1517	35610	1170	30550	41260
		Benchmarks	28863	±16553	35535	16920	0	43600
11	69000	**AngryBER$_1$**	86921	±9421	**86490**	12830	71780	**103620**
		AngryBER$_2$	**87175**	±9486	85725	18050	**74270**	101150
		Benchmarks	23159	±38533	0	53470	0	90540
12	60000	**AngryBER$_1$**	46768	±5216	**46455**	3420	**37300**	**62340**
		AngryBER$_2$	**47335**	±5644	45785	5010	36820	61730
		Benchmarks	18796	±20255	15785	35880	0	46720
13	70000	**AngryBER$_1$**	68954	±7582	70825	12270	50440	78450
		AngryBER$_2$	**73665**	±6492	**72945**	9190	**58010**	**85660**
		Benchmarks	49398	±34248	67425	71970	0	77000
14	50000	**AngryBER$_1$**	40646	±4143	41765	6000	33220	47790
		AngryBER$_2$	**46955**	±5308	**46685**	8170	**33580**	**56710**
		Benchmarks	27376	±19612	35275	43670	0	45230
15	50000	**AngryBER$_1$**	**38923**	±6883	**39325**	8880	28270	52590
		AngryBER$_2$	38473	±7922	38390	14040	**28390**	**53780**
		Benchmarks	5935	±12531	0	0	0	31120
16	62000	**AngryBER$_1$**	53034	±3972	53435	4020	44160	60280
		AngryBER$_2$	**54949**	±4099	**55245**	4480	**46450**	**66500**
		Benchmarks	5614	±17753	0	0	0	56140
17	36000	**AngryBER$_1$**	29289	±2837	28535	5120	25350	35080
		AngryBER$_2$	**29773**	±2258	**29205**	2390	**27050**	**35610**
		Benchmarks	2944	±9310	0	0	0	29440
18	60000	**AngryBER$_1$**	49648	±7015	47945	8930	39910	67960
		AngryBER$_2$	**51621**	±6740	**51020**	6810	**40400**	**69100**
		Benchmarks	9734	±20795	0	0	0	55800
19	47000	**AngryBER$_1$**	38727	±5941	39550	6030	26890	54390
		AngryBER$_2$	**42745**	±7840	**40090**	11540	**32190**	**61520**
		Benchmarks	8335	±17668	0	0	0	45580
20	52000	**AngryBER$_1$**	47751	±8239	47130	**16360**	32990	**56630**
		AngryBER$_2$	**52784**	±5267	**55150**	10	**37770**	55870
		Benchmarks	0	±0	0	0	0	0
		AngryBER$_1$	**70323**	±8395	**70500**	9910	**56890**	**90420**

offers a solution to the model selection problem by introducing sparse priors on the model parameters [14], [20], [21]. During training, the coefficients that are not significant are vanished due to the prior, thus only a few coefficients are retained in the model which are considered significant for the particular training data. This constitutes a possible direction for our future work that may improve further the proposed methodology. Finally, alternative regression mechanisms could be applied, such as Gaussian Processes, etc. [22].

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers, for their careful and detailed comments and suggestions, which have significantly improved the manuscript.

REFERENCES

[1] J. Renz, "AIBIRDS: The Angry Birds Artificial Intelligence Competition," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 4326–4327.

[2] X. Ge, S. Gould, J. Renz, S. Abeyasinghe, J. Keys, A. Wang, and P. Zhang, "Angry Birds Game Playing Software, Version 1.32, aibirds.org," Research School of Computer Science, The Australian National University, Tech. Rep., 2014.

[3] P. Zhang and J. Renz, "Qualitative Spatial Representation and Reasoning in Angry Birds: The Extended Rectangle Algebra," in *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR*, 2014.

[4] P. Walega, T. Lechowski, and M. Zawidzk, "Qualitative Physics in Angry Birds: first results," in *Symposium on Artificial Intelligence in Angry Birds*, 2014.

[5] L. A. Ferreira, G. A. W. Lopes, and P. E. Santos, "Combining qualitative spatial reasoning utility function and decision making under uncertainty on the angry birds domain," in *Symposium on Artificial Intelligence in Angry Birds*, 2013.

[6] A. G. Cohn and J. Renz, "Qualitative Spatial Reasoning," in *Handbook of Knowledge Representation*. Elsevier, 2007, pp. 551–584.

[7] P. Balbiani, J. F. Condotta, and L. F. D. Cerro, "A new tractable subclass of the rectangle algebra," in *Proceedings of the 16th International Joint Conference on Artifical Intelligence (IJCAI)*, 1999, pp. 442–447.

[8] A. Narayan-Chen, L. Xu, and J. Shavlik, "An Empirical Evaluation of Machine Learning Approaches for Angry Birds," in *Symposium on Artificial Intelligence in Angry Birds*, 2013.

[9] M. Polceanu and C. Buche, "Towards a Theory-Of-Mind-Inspired Generic Decision-Making Framework," in *Symposium on Artificial Intelligence in Angry Birds*, 2013.

[10] A. Jutzeler, M. Katanic, and J. J. Li, "Managing Luck: A Multi-Armed Bandits Meta-Agent for the Angry Birds Competition," https://aibirds.org/2013-Papers/Team-Descriptions/beaurivage.pdf, 2013.

[11] T. Borovička, R. Špetlík, and K. Rymeš, "DataLab Birds Angry Birds AI," https://aibirds.org/2014-papers/datalab-birds.pdf, 2014.

[12] J. Mendes-Moreira, C. Soares, A. Jorge, and J. F. de Sousa, "Ensemble approaches for regression: A survey," *ACM Computing Surveys*, vol. 45, no. 1, pp. 1–10, 2012.

[13] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” *Machine Learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[14] M. Tipping, “Sparse Bayesian Learning and the Relevance Vector Machine,” *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[15] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[16] N. Tziortziotis, G. Papagiannis, and K. Blekas, “AngryBER: An advanced agent for the angry birds game,” https://code.google.com/p/angry-ber/, 2014.

[17] “AIBIRDS 2014 Competition Rules,” https://aibirds.org/angry-birds-ai-competition/competition-rules.html, [Online; Accessed: 01-December-2014].

[18] “AIBIRDS 2014 Benchmarks,” https://aibirds.org/benchmarks.html, [Online; Accessed: 01-December-2014].

[19] “AIBIRDS 2014 Competition Results,” https://aibirds.org/angry-birds-ai-competition/competition-results.html, [Online; Accessed: 01-December-2014].

[20] M. Seeger, “Bayesian Inference and Optimal Design for the Sparse Linear Model,” *Journal of Machine Learning Research*, vol. 9, pp. 759–813, 2008.

[21] K. Blekas and A. Likas, “Sparse Regression Mixture Modeling with the Multi-kernel Relevance Vector Machine,” *Knowledge and Information Systems (KAIS)*, vol. 39, no. 2, pp. 241–264, 2014.

[22] C. E. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.