# Qualitative Physics in Angry Birds

Przemysław A. Wałęga, Michał Zawidzki, and Tomasz Lechowski

***Abstract*—In this paper, we present a program designed to successfully and autonomously play Angry Birds which attempts to embrace motives of human players in their choices of targets they want to shoot at in a game play. The program comprises two modules: the representation module and the reasoning module. In the former we introduce qualitative space representation that utilises notions such as "to lie on", "to lie to the right", "to be a shelter of a target", etc. The latter investigates how particular blocks of a structure behave once one of them has been hit. It includes two algorithms, namely *Vertical Impact* and *Horizontal Impact*. The first one is a novel method of investigating the behaviour of complex structures after one of their constituent blocks gets hit. Namely, it predicts which elements of a structure fall if a supporting block gets destroyed. *Horizontal Impact*, on the other hand, simulates force propagation between adjacent elements after one of them gets struck. We also describe experimental tests we have conducted in which *Vertical Impact* correctly predicted which blocks will fall in over 98% of investigated cases.**

***Index Terms*—Physics-Based Simulation Games, Qualitative Physics, Qualitative Reasoning, Spatial Reasoning, Stability Checking.**

## I. Introduction

Human reasoning about surrounding physical world is surprisingly accurate in many real life situations. In 1983 Howard Gardner presented his theory of multiple intelligences [1]. According to Gardner there are several distinct abilities that count as intelligences, one of which is the spatial intelligence. It is defined as the ability to solve problems involving navigation through space, visualization of objects from different points of view and image recognition. Spatial intelligence involves manipulation of information presented in a visual, diagrammatic or symbolic form as opposed to verbal, language based modality. We can think of spatial intelligence as a qualitative method of reasoning - it does not involve numeric values but focuses on a non-numerical presentation of the problem at hand. Application of spatial intelligence involves thinking about the shapes and arrangements of objects in space and about spatial processes, such as the deformation of objects, and the movement of objects and other entities through space.

Researchers from the field of Artificial Intelligence (AI) have been trying to construct methods that would be as effective as human reasoning about space. In some areas, like, e.g., navigation problem of car parking, contemporary AI algorithms perform better than average human (to check how precise the latter can be, see, e.g., [2]–[4]). There is, however, a large domain of problems involving spatial reasoning in which artificial methods still do not achieve as good results as humans. In particular, what we have in mind are situations that require making predictions about behaviour of objects in dynamically changing environment under uncertain or incomplete information. A good example of such a situation is a billiards game. People usually are not aware of precise numerical values of the balls' weight, stick tip's distance from the white ball or the curvature of the band, instead they use qualitative categories like "lying closely", "large angle", "hard shot" (and learn the effects of the actions taken on the basis of such notions). Still, even unproficient players who are familiar with the game can achieve reasonable scores. On the other hand, billiards robots like, e.g., Deep Green (see [5]) requires complete information about the physical environment it operates in to perform a shot.

A good platform that makes it possible to compare the performance of human and AI agents facing tasks involving reasoning about space in some simplified physical environments are the so-called Physics-Based Simulation Games (PBSG in short) like the popular Angry Birds (AB), Cut the Rope, Gears, Feed Me Oil or other. Since the physical rules are relatively simple there, these games provide a convenient testing ground for evaluating AI methods.

In this paper Angry Birds game is of our particular interest. In an AB game a player uses a slingshot to launch birds of various kinds in order to kill pigs located in a 2-dimensional physical environment and entrenched by different types of structures as depicted in Fig. 1. The goal in each level is to kill all pigs in a given game scene using only a limited number of birds of particular types. Pigs get killed whenever a sufficiently large impulse is delivered to them. Points are awarded not only for killing the pigs, but also for destroying any block appearing in the scene. In order to shoot, player needs to choose the angle at which a given bird is to be fired and the power of the shot, by simply pulling the slingshot rubber band.

Fig. 1: A screenshot from the AB game.

The main aim of this paper is to propose an approach that can successfully deal with predictive reasoning about rigid objects' behaviour upon acting forces in particular physical environments and that would employ, to a significant extent, qualitative spatial concepts known from everyday life (like, e.g., "being to the right" or "being more stable than"). The method presented in this paper is domain-specific, namely, it is

P. A. Wałęga and T. Lechowski are with the Institute of Philosophy, University of Warsaw. M. Zawidzki is with the Department of Logic, University of Łódź and the Institute of Philosophy, University of Warsaw.

Manuscript received **xxx**; revised **xxx**.

an AI agent devised to successfully play AB. The reason why we chose AB as a "running platform" for our approach is that it is a good 2-dimensional instance of such an environment and it allows for instant verification of the approach performance. Notwithstanding the fact that some elements of the method had to be tailored to this specific environment, e.g., values of particular variables (densities, shelters etc.), we believe that the techniques employed in the approach can be exploited outside the context of AB to successfully predict the rigid objects' behaviour in other similar environments. Implementing the abovementioned qualitative terms in the algorithm is supposed to bridge the gap between numeric techniques that such programs usually involve and human spatial intelligence as announced by Gardner in [1].

The main components of our approach are: a representation method and a reasoning method. The former introduces physical space representation with qualitative statements that make it possible to express such relations between objects as: "lying on", "being to the right" or "being a shelter of a target". We also introduce a method that enables us to capture the notion of "I-stability" (stability as immovableness) of an object. The second part of the approach, i.e., the reasoning method, consists of two components, namely the *Vertical Impact* (VI) and the *Horizontal Impact* (HI). The principal idea behind VI is the following: we examine the location of the center of mass of objects above our target in relation to other blocks directly below. Based on that data we infer information about the behaviour of these objects after the shot. Namely, VI decides which objects fall down after hitting a particular object. HI is based on the idea of force propagation. The force acting on the target is transferred to other blocks that are in contact with it. The transferred force depends on both the initial impact, the nature of objects involved and their relative location. HI, then, numerically estimates the impulse delivered to any object once a particular object gets hit. Our approach uses both reasoning methods to calculate the impact that an object will have on other objects upon getting hit. A numeric value of such impact is assigned to each object that is reachable by a direct shot, however we only ascribe a non-zero value to blocks which, when shot at, will affect pigs located in the scene or their sheltering structures. Ultimately, the program shoots at an object with the highest value.

The paper is organised as follows. In Sect. II we briefly discuss relevant work related to the key methods of our approach. In Sect. III we introduce the methods for representing game scenes in AB. Section IV is devoted to reasoning methods of our approach, namely VI and HI, and to calculating the final value of particular objects. An evaluation for our approach is provided in Sect. V. Finally, in Sect. VI we conclude the paper and present directions of prospective research.

## II. Related work

The work reported in this paper is mostly situated in the field of Qualitative Physics (see, e.g., [6]–[8]). The main goals of Qualitative Physics are to "produce causal accounts of physical mechanisms that are easy to understand" and at the same time are "far simpler than the classical physics and yet retain all the important distinctions without invoking the mathematics of continuously varying quantities and differential equations" as stated in de Kleer's cornerstone paper [8]. In other words, the main goal is to develop adequate representations of physical mechanisms that do not involve sophisticated mathematical tools and are understandable by the folk. The main, albeit not the only, inspiration for Qualitative Physics are our intuitions. Qualitative Physics aims at identifying the core knowledge behind our physical intuitions. Humans appear to be using qualitative representation and reasoning about the behaviour of physical environment and our everyday methods differ significantly from the classical physics' view of the world (see [8]). Given that we are good at functioning in the physical world, it is reasonable to investigate these methods.

In [7] Forbus indicates three aspects of qualitative physics: qualitative dynamics, qualitative kinematics and qualitative styles of reasoning. In our approach we traverse all of them, however, since most of the methods proposed in this paper intersect the division lines drawn by Forbus, for the sake of clarity we group them as follows. First, we provide a qualitative representation of the static state of the environment. Second, we investigate the notion of stability both in its static and dynamic aspect (the former is handled by our notion of "I-stability", the latter is done by the VI algorithm). Third, we scrutinize the way force is propagated to blocks in a structure once one of them has been hit.

*Representation:* The work concerning qualitative representation of space has been done within Qualitative Spatial Reasoning (QSR) approach (see [9]) which introduces commonsense methods representing space by means of symbolic abstraction of numerical values. Several most notable examples of QSR methods that can deal with two-dimensional space are: Region Connection Calculus [10] (although this one is more involved with topology), Allen's Interval Algebra [11] and Rectangle Algebra (RA) [12]. The Qualitative Reasoning approach has already been used in PBSG, e.g., in [13], [14] where authors introduce an extended version of RA, called Extended Rectangle Algebra (ERA). RA considers objects' Minimum Bounding Rectangles (MBRs) with sides parallel to the axes of a coordinate system (see Fig. 3). Each MBR is projected on the coordinate axes. Then, start and end points of such intervals determine a relation between two MBRs. ERA recognizes not only the start and end points of an interval but also its centre. As a result, ERA relations enable to distinguish unstable objects as it becomes apparent in the next paragraph.

*Stability:* In [15], [16] Siskind proposes a framework for kinematic analysis of stability of a two-dimensional scene composed of pentagonal blocks. He distinguishes lines that are "grounded" and then performs reasoning that answers if the whole scene is stable, whereas a scene is said to be stable if it is "immovable", i.e., cannot be moved. In Siskind's approach the reasoning is conducted for line segments, therefore polygons are represented by closed polylines. In such a scene representation some line segments intersect. Such an intersection point is called a *joint*. Siskind distinguishes three types of joints: *revolute* – if the angle between two constituent segments can change, *prismatic* – if it can "slide" along one of the constituent segments and *rigid* – otherwise. In Siskind's method each line segment should be ascribed two

numerical values representing its linear and angular velocity. A number of conditions are formulated that determine how these values should be assigned to segments if they have a common joint. The key step of Siskind's method is answering the question of whether it is possible to achieve an assignment such that it preserves certain conditions (namely such that the entropy variable introduced to the scheme can be equated to zero). If such an assignment is feasible, it means that the whole scene is movable, i.e., unstable. In other words, he investigates whether the assumption that blocks constituting the scene move is consistent, and if the answer is affirmative, he claims instability of the investigated scene. Therefore, the whole problem of stability is here reduced to solving a set of linear equations and inequalities. Ultimately, Siskind's method provides us with a yes-no answer to the question of whether a scene is stable, which does not entirely fit our needs as it will become clearer in Sect. III-D and IV-B.

In [17] Blum *et al.* analyse the notion of stability within the dynamics paradigm. The motivation of their work is to verify whether a robotic arm programmed to erect a given block construction can add subsequent blocks to the structure without losing its stability. In the cited paper also a two-dimensional variant of the problem is investigated. The method consists in analysing forces that act on particular adjacency points between blocks. Each such force occurs as a variable in a set of linear equations. Assuming that the masses of objects are known, also numbers occur in these equations. As Blum *et al.* show, the whole construction remains stable iff there exists a solution to the abovementioned set of equations in which all forces have non-negative values. The VI algorithm also "calculates" whether a gravity force acting on a block or a block structure outweighs opposite forces acting on it and cause the block (structure) fall. However, no elaborate mathematical apparatus is involved in the VI algorithm as can be seen in more detail in Sect. IV-A.

A qualitative stability analysis is delivered by Renz and Zhang in [13] and extended in [14]. In their approach an object is said to be not stable if it will move (fall) under the influence of gravity. When an object is supported appropriately, it is stable. Their definition of stable objects involves a number of rules, where the most important one seems to be the one stating that "an object is stable and will not topple if the vertical projection of the centre of mass of an object falls into the area of support base". The authors of the paper notice that the provided rules do not exhaust all cases of blocks that remain stable, which is the result of the fact that they do not take into account objects lying on the block in question, even though they could play some role in preserving stability of this block. This is caused by the fact that reasoning about supporting blocks and their influence on the object's stability only requires qualitative categories introduced by ERA, whereas if blocks lying on the object are to be taken into account, some numerical calculations would need to get involved, thus squandering the qualitative character of the whole method. Therefore it can be said that their method for reasoning about stability is sound but incomplete. The VI algorithm presented in our paper is based on intuitions about stability similar to those used in the ERA paper, however it also considers a potential impact that objects lying on an object $o$ can have on its stability.

*Force propagation:* The qualitative propagation of force (or motion) has not been extensively analysed as of today. One approach, presented by Nielsen in [18], is to consider objects arranged in the so-called kinematic chains. Two objects which are in contact with each other form a *kinematic pair*. The closure of the relation of kinematic pair is called a *kinematic chain*. If there is an external force acting on an object in a kinematic chain, this force will then be distributed to other objects in that chain. There are further questions that need to be addressed, firstly: how will a motion of an object affect another object; secondly: what kind of motion of one object will affect other objects. Nielsen describes two types of motion: translational and rotational. In order to determine how an object will move and whether it has any effect on other objects one must first eliminate the constrained motions (i.e. motions which are impossible due to some constraints, e.g., a wall) and then apply the motion transfer.

In [19] Pu considers the "flow" of force and velocity through a structure consisting of several blocks. She analyses the propagation of force for a dynamical model in terms of simple input-output mechanism. For a structure consisting of three blocks standing next to each other if we act with a certain force on the first block, it will then act with another force on the second one and so on. The incoming force is the "force-in" which is transformed into the "force-out" which in turn acts on the second block as the "force-in" again. Hence for a structure consisting of connected blocks the force flows through the structure.

In our HI algorithm we follow both of the above approaches - Nielsen's and Pu's - extending them to more general cases (e.g. when an object pushes two other objects in a given direction). For more details we refer the reader to Sect. IV-C

## III. Representation

Our approach involves spatial configuration representation by means of qualitative relations. Qualitative representation seems to be essential for a human while reasoning about space, in particular while playing PBSG games like AB. We obtain qualitative representation as an abstraction from quantitative data gathered from a screenshot of the game scene as follows.

### A. Quantitative representation

For quantitative representation of a game scene we use a two-dimensional coordinate system such that its $x$-axis is horizontal and oriented to the right, whereas the $y$-axis is vertical and oriented to the top. By $P$ we denote the set of all pixels (points) in a scene. For each

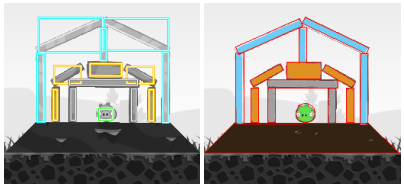


Fig. 2: Objects extracted from the AB game scene presented in Fig. 1 as Minimum Bounding Rectangles (MBRs) on the left and real shapes on the right.

pixel (point) $p$ by integers $x(p)$ and $y(p)$ we denote its $x$- and $y$-coordinate respectively. We obtain a list of objects $O$ in the game scene by means of a basic vision module provided by the Angry Birds AI Competition organizers (available online at aibirds.org) as presented in Fig. 2. Each object extracted from the game scene screenshot is represented as a set of points. Henceforth, if it does not lead to confusion, we interchangeably use $o$ as denoting an object and the set of its constituent points. For any given object $o$, $center(o)$ is its center point, $area(o)$ is its area, $width(o)$ and $height(o)$ are dimensions of a Minimum Bounding Rectangle (MBR) of $o$, and $contour(o)$ is the set of borderline points that belong to $o$, as presented in Fig. 3. Additionally, objects have assigned types, namely ice, wood, pig or stone, e.g., $pig(o)$ denotes that $o$ is a pig. Then, we define a density function, $\rho : \{ice, wood, stone, pig\} \rightarrow [0, 1]$ which ascribes to each type of object an experimentally obtained value of density ($\rho(ice) = 0.6$, $\rho(wood) = 0.7$, $\rho(stone) = 1$, $\rho(pig) = 0.6$)[1], which we use to compute mass of an object. Furthermore there is one distinguished object, namely $ground$.

For any two objects $o_i$ and $o_j$ the distance $dist(o_i, o_j)$ is the smallest distance between points $p_k$, $p_l$ such that $p_k \in o_i$ and $p_l \in o_j$. Additionally, by $traj(o) = \{traj_{\rightarrow}(o), traj_{\nearrow}(o)\}$, we denote a set of 2 parabolic trajectories (low and high) of a shot targeted at the top left point of $o$ with the greatest possible sling tension,

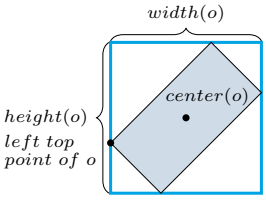


Fig. 3: Object's $o$ representation.

where $traj_{\rightarrow}(o)$ ($traj_{\nearrow}(o)$) is a list of points belonging to the low (high) trajectory (see Fig. 6), such that its first point is the central point of the sling and the last point is the left top point of $o$ (in most cases in AB shooting at the left top point of an object gives better results, then shooting at its center). Trajectories ignore obstacles, i.e., when determining a trajectory between the sling and an object we abstract from the fact that it might intersect other objects in between. Obviously, in some cases there is only one or even no possible trajectory. In what follows, the list of trajectories calculated for a given game scene is denoted by $T$. The vision module enables us to determine the bird type that is going to be launched. We distinguish the following bird types: red, blue, yellow, white and black. A list of all possible bird types is denoted by $B$.

We introduce the $reachable$ predicate for objects. Intuitively, it expresses the fact that an object $o$ may be directly hit by a shot. An object $o$ is $reachable$ iff there exists a trajectory $t$ (low or high) aiming at $o$ and not intersecting any object $o_k$ such that $o_k \neq o$. More formally:

**Definition 1** ($reachable$ predicate)**.** For an object $o \in O$

$$reachable(o) \equiv \exists t \in traj(o) \forall p \in t \forall o_i \in O \big((p \in o_i) \rightarrow (o_i = o)\big).$$

[1]At this stage of development of the program, the algorithm is not faced with incomplete information like, e.g., unknown numerical values of densities. Instead, we rigidly "fill out" these gaps. Introducing learning module to the algorithm is a matter of future research (see Sect. VI).

### B. Qualitative representation

We define $lies\_on$ - a binary relation between objects. Intuitively, $lies\_on(o_1, o_2)$ whenever $o_1$ is adherent to $o_2$ from above and presses $o_2$ with its weight. Definition 2 captures this intuition more formally:

**Definition 2** ($lies\_on$ relation)**.** For objects $o_1, o_2 \in O$

$$lies\_on(o_1, o_2) \equiv o_1 \neq o_2 \wedge \exists p_1, p_2 \in P\big(p_1 \in o_1 \wedge p_2 \in o_2 \wedge x(p_1) = x(p_2) \wedge 0 < y(p_1) - y(p_2) < c\big).$$

The meaning of the constant $c \in \mathbb{N}$ is the following. Normally we would be looking for only those points of $o_1$ that strictly adhere to $o_2$ as the candidates for the leftmost and rightmost points (in that case $c = 1$). However, due to imperfect representation of blocks provided by the vision module it might happen that two objects that are adherent in the actual game scene are represented as slightly distant to each other. Therefore, the aim of introducing the constant $c$ is to prevent recognizing such imperfectly represented objects as disconnected.

The $lies\_on$ relation is irreflexive and for convex polygons (which we are mostly dealing with in the AB game scenes) also asymmetric (this asymmetry breaks down in the case of non-convex polygons). At first glimpse $lies\_on$ resembles the intuitive concept of "being supported by" and one could be inclined to confuse these two. It is not the case, though, since the latter is not asymmetric even for convex polygons. For instance, in Fig. 4(a) we have $lies\_on(o_1, o_2)$ but at the same time $o_1$ is a support of $o_2$ as well as $o_2$ is a support of $o_1$.

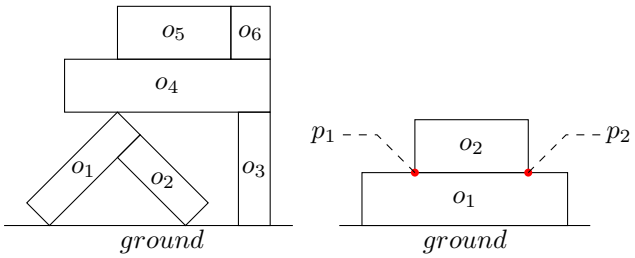


Fig. 4: In Fig. (a) a sample arrangement of blocks is presented. Figure (b) shows the leftmost (rightmost) connection points $p_1$ ($p_2$) between $o_1$ and $o_2$.

Analogously, $on\_right(o_1, o_2)$ (also irreflexive and asymmetric for convex polygons) means that $o_1$ adheres to $o_2$ from the right-hand side stabilizing $o_1$ in the event it gets hit by (or after) a shot (as it will become apparent later, forces are generally propagated from left to right). For example, in Fig. 4(a) we have $on\_right(o_6, o_5)$. A formal definition of $on\_right$ is as follows.

**Definition 3** ($on\_right$ relation)**.** For objects $o_1, o_2 \in O$

$$on\_right(o_1, o_2) \equiv o_1 \neq o_2 \wedge \exists p_1, p_2 \in P\big(p_1 \in o_1 \wedge p_2 \in o_2 \wedge y(p_1) = y(p_2) \wedge 0 < x(p_1) - x(p_2) < c\big).$$

Henceforth we abbreviate $lies\_on^{-1}$ and $on\_right^{-1}$ by, respectively, $lies\_under$ and $on\_left$, i.e., $lies\_under$ ($on\_left$) is the converse relation of $lies\_on$ ($on\_right$).

For each two blocks $o_1, o_2$ such that $lies\_on(o_1, o_2)$, we define the left- and rightmost connection points between $o_1$ and $o_2$ ($left\_p(o_1, o_2)$ and $right\_p(o_1, o_2)$, respectively) in 3 steps. First, let $L(o_1, o_2)$ be a set of all points $p \in o_1$ such that their distance to $o_2$ is smaller than the constant $c$. Second, let $L'(o_1, o_2) \subseteq L(o_1, o_2)$ be a set of points with the smallest $x$-coordinate. Third, the $left\_p(o_1, o_2)$ is the point in $L'(o_1, o_2)$ with the smallest $y$-coordinate. We define the $right\_p$ in an analogous way. As an example consider the structure presented in Fig. 4(b), where $lies\_on(o_1, o_2)$, hence we can determine the $left\_p(o_1, o_2)$ and $right\_p(o_1, o_2)$. In this case we have $p_1 = left\_p(o_1, o_2)$ and $p_2 = right\_p(o_1, o_2)$.

The main part of spatial representation is obtained by means of the unary predicate $reachable$ and the binary relations $lies\_on$ and $on\_right$. In what follows we introduce a spatial network – a graph-like structure which is constructed and stored by our program.

**Definition 4** (spatial network)**.** *Spatial network* $G = (O, reachable, lies\_on, on\_right)$ is a graph-like structure, where $O$ is a set of spatial objects, $reachable \subseteq O$ is a set of distinguished (reachable) objects, and $lies\_on \subseteq O^2$ and $on\_right \subseteq O^2$ constitute a set of edges of two types.

As an example we show in Fig. 5(a) a graphical presentation of a spatial network corresponding to the block structure from Fig. 4(a).

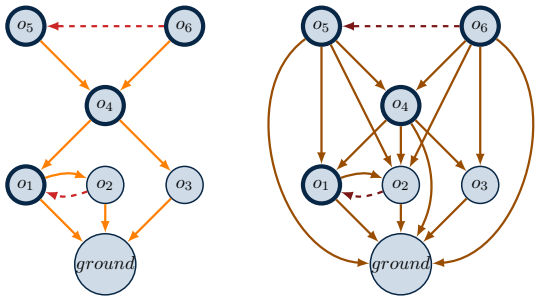



Fig. 5: The graph-like representation of the structure from Fig. 4(a). (a) represents a spatial network, where thick nodes are elements of $reachable$, orange arrows represent $lies\_on$ relation and red dashed arrows stand for $on\_right$. In (b) respective shaded arrows represent $lies\_on^*$ and $on\_right^*$.

We denote the transitive closure[2] of the relations $lies\_on$ and $on\_right$ by $lies\_on^\star$ and $on\_right^\star$, respectively. We have obtained transitive closures of $lies\_on$ and $on\_right$ by means of a depth-first search algorithm presented in [20] (whose complexity is in $\mathcal{O}(n^2)$, where $n = card(V)$ is a number of nodes in the graph). The obtained $lies\_on^\star$ and $on\_right^\star$ enable us to create a graph-like structure $G' = (O, reachable, lies\_on^\star, on\_right^\star)$ presented in Fig. 5(b).

### C. Shelters

It often appears that there is no way of hitting a target without destroying its sheltering structure first. It is then reasonable to identify the most important shelters. We define $shelter$ - a ternary relation between two objects and a trajectory, which denotes the fact that the first object is a shelter for the second object (always a pig) with respect to a given trajectory. More precisely, $shelter(o_1, o_2, t)$ whenever $o_2$ is a pig and $o_1$ lies on one of estimated trajectories for a shot aiming at $o_2$. A formal definition of the $shelter$ relation is as follows.

**Definition 5** ($shelter$ relation)**.** For objects $o_1, o_2 \in O$ and a trajectory $t \in T$

$$shelter(o_1, o_2, t) \equiv t \in traj(o_2) \wedge pig(o_2) \wedge o_1 \cap t \neq \emptyset.$$

For each object we determine a shelter value which allows us to distinguish between more and less important shelters. We define a function $shelter\_val : O \to \mathbb{R}$ that maps an object into its shelter value as follows. Let $T_{pig} \subseteq T$ be a set of all trajectories that aim at any of the pigs in a game scene and let $T_{pig}(o) \subseteq T_{pig}$ be a set of all these trajectories that intersect $o$ and do not directly aim at $o$, i.e., the trajectories with respect to which $o$ is a shelter for some pig. Now, the more elements $T_{pig}(o)$ contains, the more valuable $o$ should be as a shelter. Moreover, the closer a sheltering object $o$ is to the pig and the fewer objects lie between $o$ and this pig on a given trajectory, the higher the probability that a direct shot at $o$ will also affect the pig and, consequently, the higher the shelter value of $o$ should be.

**Definition 6** ($shelter\_val$ function)**.** For an object $o \in O$

$$shelter\_val(o) = \sum_{t \in T_{pig}(o)} \frac{1}{dist(o, pig_t)} \cdot \frac{1}{no\_betw(o, pig_t, t) + 1},$$

where $pig_t$ is a pig that a trajectory $t$ aims at and $no\_betw(o, pig_t, t)$ is a number of objects lying on a trajectory $t$ between $o$ and $pig_t$ .

Note that the intuitive meaning of the $shelter\_val$ function is a little vague and comprises the importance of a shelter from the player's point of view, i.e., to what extent destroying it will help them in further game play.

As an example consider Fig. 6. On a trajectory $traj_{\rightarrow}(o_4)$ there are 2 shelters for the $o_4$ pig, namely $o_2$ and $o_3$, whereas on the upper trajectory $traj_{\nearrow}(o_4)$ only $o_3$ is a shelter for $o_4$. Consequently, $T_{pig}(o_1) = \emptyset$, $T_{pig}(o_2) = \{traj_{\rightarrow}(o_4)\}$, $T_{pig}(o_3) = \{traj_{\rightarrow}(o_4), traj_{\nearrow}(o_4)\}$. $o_3$ has higher $shelter\_val$ than $o_2$ since, first, $|T_{pig}(o_3)| > |T_{pig}(o_2)|$, second, $o_3$ is closer to the $o_4$ pig with respect to trajectory $traj_{\rightarrow}(o_4)$ and last, there are no other objects between $o_3$ and $o_4$ on either of the trajectories intersecting $o_3$ (as opposed to $o_2$).

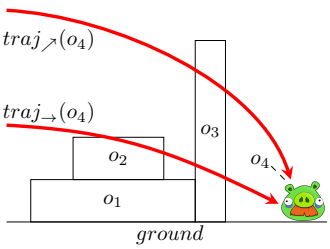



Fig. 6: For the $traj_{\rightarrow}(o_4)$ trajectory $o_2$ and $o_3$ are shelters of the $o_4$ pig, whereas for $traj_{\nearrow}(o_4)$ only $o_3$ is a shelter of $o_4$.

[2]The transitive closure of a binary relation $R$ on a set $X$ is a minimal transitive relation $R^*$ on $X$ such that $R \subseteq R^*$.

### D. I-stability

In our reasoning module we employ the notion of I-stability (stability as immovableness) that would be helpful in determining whether an object is likely to fall/move after getting hit. Since we want to decide which objects are the best targets, i.e., which should we shoot at to cause the biggest possible damage, we need I-stability to be a gradable property. Thus, we define I-stability as "susceptibility to fall or move upon getting hit". It is easy to note that neither of the stability definitions presented in Sect. II meets the above case. Following the lines of [6], [7], we can informally characterize the introduced notion by indicating certain monotonic dependencies between I-stability and various features of a game scene. We will use Fig. 7 as an example. Thus, the object $o$'s I-stability increases[3] whenever one of the following increases (ceteris paribus):

- $n$ - I-stability of objects that $lie\_under$ $o$ ($o_2$, $o_3$)
- $ratio(o)$ - ratio of $o$'s width to its height ($ratio(o_1)$)
- $mass(o)$ - cumulated mass of $o$ and all objects that $lie\_on^*$ $o$ (mass of $o_1$, $o_4$ and $o_5$),
- $mass\_right(o)$ - cumulated mass of the objects that are directly $on\_right$ of $o$ (mass of $o_6$ and $o_7$).

Before we briefly comment on each of these points, we introduce a formal definition of a function $I\text{-}stability : O \rightarrow [0,1]$ that assigns to each object an I-stability value from the interval $[0,1]$ (where 0 means that the object will move without any stimuli and 1 means that the object is impossible to move).

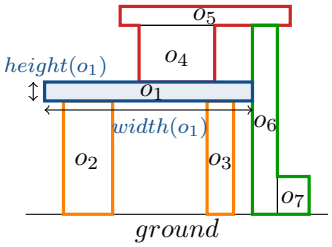



Fig. 7: Factors determining the I-stability of $o$.

**Definition 7** ($I\text{-}stability$ function)**.** For an object $o \in O \setminus \{ground\}$

$$I\text{-}stability(o) = \left( \sqrt[n]{\frac{1}{\sum\limits_{o_i | lies\_on(o,o_i)} \frac{1}{I\text{-}stability(o_i)}}} \cdot ratio(o) \right)^{\sqrt[k]{mass(o)} \cdot \sqrt[l]{mass\_right(o)}},$$

where:

$$ratio(o) = \begin{cases} 1 & \text{if } width(o) \geq height(o) \\ \frac{width(o)}{height(o)} & \text{otherwise} \end{cases},$$

$$mass(o) = \frac{1}{1 + \sum\limits_{o_i | o_i = o \vee lies\_on^*(o_i,o)} area(o_i)\rho(o_i)},$$

$$mass\_right(o) = \frac{1}{1 + \sum\limits_{o_i | on\_right^*(o_i,o)} area(o_i)\rho(o_i)},$$

$n = \mathsf{card}(\{o_2 \mid lies\_on(o_1, o_2)\})$,

$k, l \in \mathbb{Z}$ are determined experimentally.

I-stability of $ground$ is always equal to 1.

[3]Until it reaches the threshold value here equal to 1.

Notice that a value of the $I\text{-}stability$ function is always in the interval $[0,1]$. $\frac{1}{\sum\limits_{o_2 | lies\_on(o_1,o_2)} \frac{1}{I\text{-}stability(o_2)}}$ is in the interval $[0,1]$ and $n \in \mathbb{N}$, therefore $\sqrt[n]{\frac{1}{\sum\limits_{o_2 | lies\_on(o_1,o_2)} \frac{1}{I\text{-}stability(o_2)}}} \in [0,1]$ and the greater $n$ is, the greater is this expression. Since $ratio(o_1) \in [0,1]$, then $\left( \sqrt[n]{\frac{1}{\sum\limits_{o_2 | lies\_on(o_1,o_2)} \frac{1}{I\text{-}stability(o_2)}}} \cdot ratio(o_1) \right) \in [0,1]$. Finally, raising this expression to any non-negative power cannot result in getting a value greater than 1.

Now let us get back to the monotonic dependencies characterizing our notion of I-stability. The first one expresses the fact that an object "inherits" some part of its I-stability after the objects lying directly under. In our equation we used a variation of geometric average to encode that if more than one block underlies an object under consideration then the more I-stable of them affect the "inherited" part of I-stability of the object stronger than less I-stable ones. Even though this assumption is not free of fallacies, we believe it reflects well an "average" case. The second dependency reflects an intuitive truth that the "flatter" an object is, the harder it is to move or overthrow. In the equation the ratio value of an object is used as a simple coefficient. The third dependency encapsulates the fact that mass pressing an object from above stabilizes it. The last one captures a simple relation: the more massive is a structure situated to the right of an object, the less the object is susceptible to move or fall after getting hit from the right (since it can abut against this structure). Both mass values are captured in the equation in the form of exponents. Since any non-zero value of a mass equates the value of the respective exponent to a fraction from the interval $(0,1)$, it increases the value of the base (which is a fraction from an interval $(0,1]$)[4]. The roots in the exponents and their degrees (represented by $k$ and $l$ in the equation) are introduced to balance the influence of both types of mass on the final I-stability value of an object. In the case of our algorithm $k$ and $l$ have been experimentally determined to be equal to, respectively, 5 and 7.

At the end of this section let us again recall other definitions of stability discussed already in Sect. II. Let us consider the objects $o_6$ and $o_7$ from Fig. 7. According to [14] both objects are stable as standing on the ground (satisfying Rule 1). In the light of [17], too, both of them would be classified as stable since all forces acting on them remain in equilibrium. On the other hand, Siskind's method from [15], [16] would qualify them as unstable since neither of them is grounded (only the ground itself and hills are "grounded" in Siskind's sense) and it is possible to move both of them. Our approach assigns different I-stability values to both objects determining which is more likely to overturn after getting hit. It shows that the notion of I-stability introduced in this paper recognizes a different property than the cited works.

## IV. Reasoning

In this section, we demonstrate how we use the notions constituting our representation of a game scene in order to

[4]Note that if $mass\_above(o)$ or $mass\_right(o)$ is equal to 0, then the respective exponent is equal to 1 thus leaving the base unchanged.

take the best possible shot, i.e., the shot that, given the circumstances, will cause the largest damage to pigs and/or their shelters in the scene. The aim of reasoning methods displayed in this section is to assign numerical values to objects, so that, in principle, an object with the highest value is the best direct target in a given situation. Calculation of the final values of objects consists of the following parts:

**First**, we assume that the only objects that we want to have impact on are pigs and their shelters.

**Second**, we indicate all objects that are reachable by a shot (for details, see Sect. III-A) since these are the only objects that we are going to assign a non-zero value to.

**Third**, in calculating the final value of each reachable block we take into account two components: its I-stability and the degree to which violation of the object will have impact on pigs and their shelters. The pattern appropriately embracing these factors is presented in the final part of this section.

**Fourth**, in calculating the impact of a block on a pig or its shelter, we take into account two factors, namely: which blocks will fall after we have damaged a given block, and how the impact of the shot is propagated into other blocks. We introduce two methods for capturing the abovementioned factors, namely *Vertical Impact* and *Horizontal Impact* respectively. They are presented in detail in the forthcoming two parts of this section.

### A. Vertical Impact

In most Angry Birds scenes objects are arranged into complex structures. This means that by striking one of them we also affect other components of the structure. In such structures some objects support others, so that if we remove these supporting blocks (they get destroyed or fall), supported objects may collapse. We propose a method which is qualitative in nature and only to a small extent exploits quantitative measures. We call it *Vertical Impact*. It allows us to infer about the effects of an object's fall or disappearance, i.e., predicting which other blocks in the structure will fall. The method is based on a recursive checking of substructures in the spatial network representing the scene (starting from the object directly hit by a bird) and in the end it returns the list of all objects that ultimately will fall. It exploits the rule that serves as the basis for stability investigation in [14], namely that "an object is stable and will not topple if the vertical projection of the centre of mass of an object falls into the area of support base". In the following part we demonstrate in detail how the method works, using Fig. 8 as our running example.

We apply VI by taking as an input a concrete object $o$ we intend to destroy by a direct shot (see Fig. 8(a)). Afterwards, we proceed in the following steps.

**First**, we determine a set $A_o$ of all objects in the relation $lies\_on^*$ with $o$, i.e., lying directly or indirectly on $o$[5]. Afterwards we find a set $B_o$ of all objects lying under $\mathfrak{A}_o$ except for $o$ itself. Intuitively, after the fall of $o$ $\mathfrak{A}_o$ rests on objects from $B_o$, which form a structure henceforth referred to as $\mathfrak{B}_o$. In Fig. 8(b) $A_{o_1} = \{o_3, o_4, o_5, o_8\}$ (marked blue) and $B_{o_1} = \{o_2, o_6\}$ (marked orange).

[5]Henceforth, we will denote an object composed of all elements of a set $S$ by a corresponding symbol $\mathfrak{S}$ written in gothic font. For example, $\mathfrak{A}_o$ will denote an object built of all elements of $A_o$.

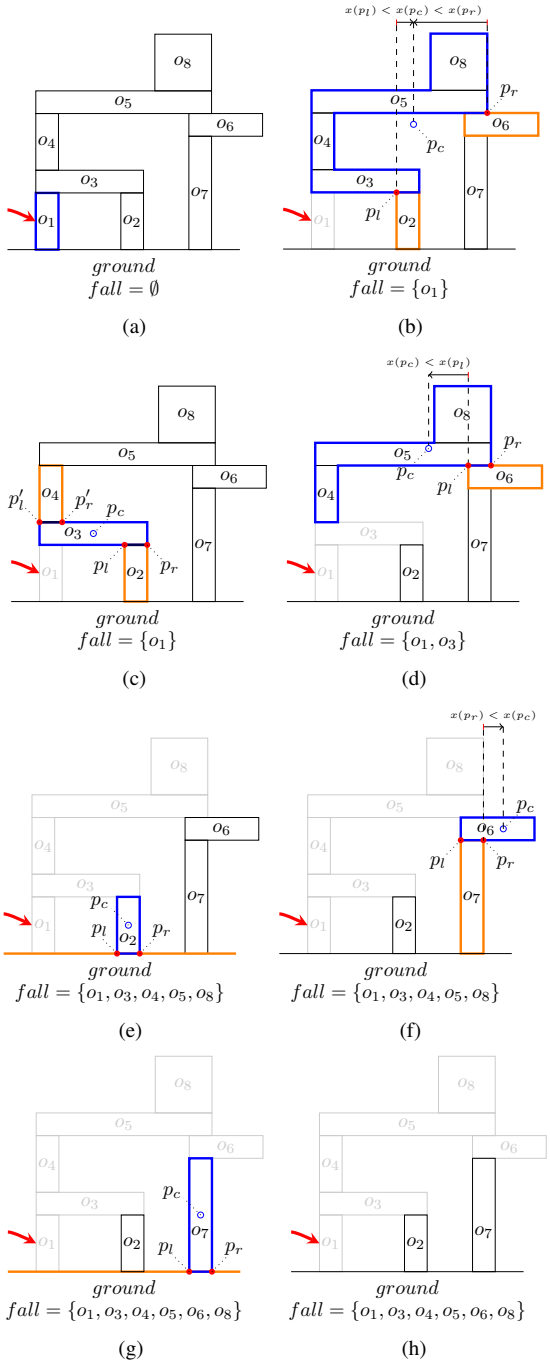



Fig. 8: Blue lines circle elements that are currently processed by the VI algorithm. Elements marked orange lie directly under (or also directly on when the *Check* procedure is launched) objects circled with a blue line.

**Second**, we fix three points:

1) the center of mass for $\mathfrak{A}_o$ ($center(\mathfrak{A}_o)$, $p_c$ in short),
2) the leftmost connection point between $\mathfrak{A}_o$ and $\mathfrak{B}_o$ ($left\_p(\mathfrak{A}_o, \mathfrak{B}_o)$, $l_p$ in short),
3) the rightmost connection point between $\mathfrak{A}_o$ and $\mathfrak{B}_o$ ($right\_p(\mathfrak{A}_o, \mathfrak{B}_o)$, $r_p$ in short),

We call the last two the *pivot points* of $\mathfrak{A}_o$.

**Third**, we check whether $x(p_c) < x(p_l)$ or $x(p_c) > x(p_r)$. If neither is the case, we proceed to step **Fourth**. Otherwise, we perform Step **Sixth**. Fig. 8(b) shows that the $\mathfrak{A}_{o_1}$'s centre of mass $p_c$ is located between the left- and rightmost connection point ($p_l$ and $p_r$, respectively) between $\mathfrak{A}_{o_1}$ and $\mathfrak{B}_{o_1}$, so we need to perform a further check whether $\mathfrak{A}_{o_1}$ will indeed remain stable.

**Fourth**, If $x(p_l) < x(p_c) < x(p_r)$, in most cases it means that $\mathfrak{A}_o$ remains stable and nothing is added to the list of fallen objects at this stage. However, it might also turn out that even though the $\mathfrak{A}_o$ as a single object would remain stable, the stability of some components of $A_o$ has been violated. To verify this, we check, one by one, each block lying on our object $o$. For each such object $o_i$ we fix two sets: $LO_{o_i}$ of all objects lying directly on $o_i$ and $LU_{o_i}$ (to memorize these symbols, note that $LO$ and $LU$ stand for, respectively, *lying on* and *lying under*) of all objects lying directly under $o_i$. If $LU_{o_i}$ turns out to be empty, it means that $o_i$ has no longer any blocks it can lie on and it falls. Therefore, we launch the VI method for $o_i$ as an input and for the initial *spatial network* reduced by the set of objects currently occurring on the list of fallen objects and the set of respective edges departing or entering these objects. In Fig. 8(c) the only object that has been lying directly on $o_1$ is $o_3$. $LU_{o_3}$ is non-empty and consists of a single object $o_2$, while $LO_{o_3} = \{o_4\}$.

**Fifth**, If $LU_{o_i}$ is not empty, which means that there exist blocks that $o_i$ is lying on, it is still possible that $o_i$ has lost its stability. We check it by, first, fixing the following points:

1) the left- and rightmost connection points between $\mathfrak{LU}_{o_i}$ and $o_i$ ($left\_p(\mathfrak{LU}_{o_i}, o_i)$, $right\_p(\mathfrak{LU}_{o_i}, o_i)$, $p_l$ and $p_r$, respectively, in short),
2) the left- and rightmost connection points between $\mathfrak{LO}_{o_i}$ and $o_i$ ($left\_p(\mathfrak{LO}_{o_i}, o_i)$, $right\_p(\mathfrak{LO}_{o_i}, o_i)$, $p'_l$ and $p'_r$, respectively, in short),
3) the centre of mass of $o_i$ ($center(o_i)$, $p_c$ in short).

Now, if both the $p_c$ and the entire adherent part of $\mathfrak{LO}_{o_i}$ protrude beyond the (entire) adherent part of $\mathfrak{LU}_{o_i}$ from the same side, i.e., $x(p_c) < x(p_l) \wedge x(p'_r) < x(p_l)$ or $x(p_r) < x(p_c) \wedge x(p_r) < x(p'_l)$, it means that there is no counterbalance for the forces acting on $o_i$ on this side from above. Consequently, $o_i$ looses its stability and falls. Thus, we launch the VI method for $o_i$ as an input and for the initial *spatial network* reduced by the set of objects currently occurring on the list of fallen objects and the set of respective edges departing or entering these objects. In Fig. 8(c) we can see that both the $o_3$'s centre of mass $p_c$ and the whole part of $\mathfrak{LO}_{o_3}$ adherent to $o_3$ (determined by the interval $[p'_l, p'_r]$) protrude from the left beyond the part of $\mathfrak{LU}_{o_3}$ adherent to $o_3$ (determined by the interval $[p_l, p_r]$), which means that $o_3$ will fall and is added to the list $fall$.

**Sixth** If $x(center(\mathfrak{A}_o)) < x(left\_p(\mathfrak{A}_o, \mathfrak{B}_o))$ or $x(center(\mathfrak{A}_o)) > x(right\_p(\mathfrak{A}_o, \mathfrak{B}_o))$, $\mathfrak{A}_o$ must have lost its balance and in effect all elements $A_o$ fall. Consequently, we add them to the list of fallen objects. Roughly, if a centre of mass of the structure $\mathfrak{A}_o$ is located horizontally between utmost connection points between $\mathfrak{A}_o$ and its underlying structure $\mathfrak{B}_o$, in most cases it remains stable (unless the *Check* procedure from steps **Fourth** and **Fifth** proved the opposite). On the other hand, if it protrudes from the left beyond the leftmost connection point or from the right beyond the rightmost connection point, $\mathfrak{A}_o$ loses its balance and falls. In Fig. 8(d) the sets $A_{o_3} = \{o_4, o_5, o_8\}$ and $B_{o_3} = \{o_6\}$ are determined since $o_3$ has just fallen. It turns out that $\mathfrak{A}_{o_3}$'s centre of mass protrudes from the left beyond $\mathfrak{B}_{o_3}$. Consequently, all elements of $A_{o_3}$ fall and are added to the list $fall$.

**Algorithm 1** VERTICAL($o$)

**Input:** an object $o$
**Output:** a list $fall$ of all objects that fall if $o$ does

Initialize an empty list of objects $fall$;
Run FALL($o$, $fall$);

**Algorithm 2** FALL($o$, $fall$)

**Input:** an object $o$ and an initialized list $fall$
**Output:** an updated list $fall$

Add $o$ to $fall$;
Set a list of objects $A_o \leftarrow \{o' \in O \mid lies\_on^*(o', o) \wedge o' \notin fall\}$;
Initialize a new object $\mathfrak{A}_o$;
Set $contour(\mathfrak{A}_o) \leftarrow contour(\bigcup A_o)$; // $\mathfrak{A}_o$ takes its contour after the contour of $A_o$ understood mereologically
Initialize a double $x(\mathfrak{A}_o)$; // an abscissa of $\mathfrak{A}_o$'s mass center
Set a double $m \leftarrow 0$ // $m$ will represent mass of $\mathfrak{A}_o$
**for each** object $o_i$ from $A_o$ **do**
  Set $m \leftarrow m + area(o_i) \cdot \rho(o_i)$; // $area(o_i) \cdot \rho(o_i)$ is a mass of $o_i$
**end for**
**for each** object $o_i$ from $A_o$ **do**
  Set $x(\mathfrak{A}_o) \leftarrow x(\mathfrak{A}_o) + \frac{area(o_i)\rho(o_i)}{m} \cdot x(o_i)$; // calculating $x(\mathfrak{A}_o)$ as the weighted average of the abscissas of all $A_o$'s members' centers
**end for**
Set lists of objects $B_o \leftarrow \{o' \in O \mid o' \notin fall \cup A_o \wedge \exists o'' \in A_o\ lies\_under(o', o'')\}$, $B_o^+ \leftarrow B_o \cup \{o' \in O \mid lies\_under(o', o) \wedge o' \notin fall\}$;
**if** $B_o$ is empty **then** // if $o$ turned out to be the only block lying under $\mathfrak{A}_o$
  Set $fall \leftarrow fall \cup \bigcup A_o$;
  **for each** object $o_i$ from $B_o^+$ **do**
    Set a list $C_{o_i} \leftarrow \{o' \in O \mid lies\_under(o', o_i)\}$;
    **if** $C_{o_i}$ is not empty **then**
      Initialize a new object $\mathfrak{C}_{o_i}$;
      Set $contour(\mathfrak{C}_{o_i}) \leftarrow contour(\bigcup C_{o_i})$;
      **if** not $left\_p(o_i, \mathfrak{C}_{o_i}) < x(o_i) < right\_point(o_i, \mathfrak{C}_{o_i})$ **then**
        Run FALL($o_i$, $fall$);
      **end if**
    **end if**
  **end for**
**else**
  Initialize a new object $\mathfrak{B}_o$;
  $contour(\mathfrak{B}_o) \leftarrow contour(\bigcup B_o)$;
  **if** $left\_p(\mathfrak{A}_o, \mathfrak{B}_o) < x(\mathfrak{A}_o) < right\_p(\mathfrak{A}_o, \mathfrak{B}_o)$ **then** // if the $\mathfrak{A}_o$' mass center lies between both utmost connection points between $\mathfrak{A}_o$ and the structure $\mathfrak{B}_o$ lying under $\mathfrak{A}_o$
    **for each** object $o_i$ such that $lies\_on(o_i, o)$ **do**
      Run CHECK($o_i$, $fall$);
    **end for**
  **else**
    Set $fall \leftarrow fall \cup \bigcup A_o$;
    **for each** object $o_i$ from $B_o^+$ **do**
      Set a list $C_{o_i} \leftarrow \{o' \in O \mid lies\_under(o', o_i)\}$;
      **if** $C_{o_i}$ is not empty **then**
        Initialize a new object $\mathfrak{C}_{o_i}$;
        Set $contour(\mathfrak{C}_{o_i}) \leftarrow contour(\bigcup C_{o_i})$;
        **if** not $left\_p(o_i, \mathfrak{C}_{o_i}) < x(o_i) < right\_point(o_i, \mathfrak{C}_{o_i})$ **then**
          Run FALL($o_i$, $fall$);
        **end if**
      **end if**
    **end for**
  **end if**
**end if**

**Algorithm 3** CHECK($o, fall$)

**Input:** an object $o$ and an initialized list $fall$
**Output:** an updated list $fall$

1: Set lists $LO_o \leftarrow \{o' \in O \mid lies\_on(o', o)\}$, $LU_o \leftarrow \{o' \in O \mid lies\_under(o', o)\}$;
2: **if** $LU_o$ is empty **then**
3: | Run FALL($o, fall$);
4: **else**
5: | Initialize new objects $\mathfrak{OO}_o$, $\mathfrak{UU}_o$;
6: | Set $contour(\mathfrak{OO}_o) \leftarrow contour(\bigcup LO_o)$, $contour(\mathfrak{UU}_o) \leftarrow contour(\bigcup LU_o)$;
7: | **if** $[right\_p(\mathfrak{OO}_o, o) < left\_p(\mathfrak{UU}_o, o)$ & $x(center(o)) < left\_p(\mathfrak{UU}_o, o)]$ **or** $[left\_p(\mathfrak{OO}_o, o) > right\_p(\mathfrak{UU}_o, o)$ & $x(center(o)) > right\_p(\mathfrak{UU}_o, o)]$ **then** // if two conditions are jointly satisfied: 1) $o$'s mass center protrudes beyond the adherent structure $\mathfrak{UU}_o$ lying under $o$ from the left (right), 2) the adherent structure $\mathfrak{OO}_o$ pressing $o$ from above also protrudes beyond $\mathfrak{UU}_o$ from the left (right)
8: | | Run FALL($o, fall$);
9: | **end if**
10: **end if**

**Seventh**, we begin with constructing a set $B_o^+$ consisting of the elements of $B_o$ and all objects lying directly under $o$. If $\mathfrak{A}_o$ has not collapsed, we further consider only the objects located directly under $o$. Otherwise we consider all elements of $B_o^+$. For all these elements we check whether they fall by taking the following steps. We fix a set $C_{b_i}$ of all objects lying directly under the object $b_i$. They form a structure which we denote as $\mathfrak{C}_{b_i}$. Then we fix three points:

1) the center of mass for the object $b_i$ ($center(b_i)$, $p_c$ in short),
2) the leftmost connection point between $b_i$ and $\mathfrak{C}_{b_i}$ ($left\_p(b_i, \mathfrak{C}_{b_i})$, $p_l$ in short),
3) the rightmost connection point between $b_i$ and $\mathfrak{C}_{b_i}$ ($right\_p(b_i, \mathfrak{C}_{b_i})$, $p_r$ in short).

If $x(p_c) < x(p_l)$ or $x(p_r) < x(p_c)$, then $b_i$ falls and we launch the VI method for $b_i$ as an input and for the initial *spatial network* reduced by the set of objects currently occurring on the list of fallen objects and the set of respective edges departing or entering these objects (since we assume that, as fallen, they no longer take part in the reasoning process). Otherwise, it remains stable so we proceed to the next $B_o$'s element $b_{i+1}$ and we get back to the previous point of the method. In Fig. 8(e) we can see that $B_{o_3} = \{o_6\}$ is extended to $B_{o_3}^+$ by adding $o_2$ as the only element that has been lying directly under $o_3$. Thus, we proceed to checking whether subsequent elements of $B_{o_3}^+ = \{o_2, o_6\}$ will fall. The first element of $B_{o_3}^+$ $o_2$ lies on the $ground$ which obviously makes it remain stable (see Fig. 8(e)). The second element of $B_{o_1}^+$ element, $o_6$ falls since its center of mass $p_{cg}$ protrudes to the right the rightmost connection point $p_{rm}$ between itself and the only element of $C_{o_6}$, i.e., $o_7$ (see Fig. 8(f)). Since $o_6$ has fallen, we launch VI with $o_6$ as an input and an appropriately shrunk *spatial network*. There is no object that lies on $o_6$, so we instantly proceed to checking elements of $B_{o_6}^+$ which is a singleton set $\{o_7\}$ (see Fig. 8(g)). However, $o_7$ remains stable, so the algorithm terminates returning the list $fall = \{o_1, o_3, o_4, o_5, o_6, o_8\}$ of all fallen objects (see Fig. 8(h)).

A pseudocode of the VI method is demonstrated in Algorithms 1 (*Vertical*), 2 (*Fall*) and 3 (*Check*). Intuitively, Algorithm 1 serves to initialize a list $fall$ that is to be filled. Algorithm 2 makes it possible to verify whether whole structures lying transitively on a fallen object will fall. Algorithm 3 enables us to check if particular elements of a structure that Algorithm 2 evaluated as stable will indeed remain stable.

The VI method always terminates. Indeed, by the fact that the number of objects in a game scene is limited, the *Fall* method can be recursively called only a finite number of times - it cannot be called for the same object twice, which is secured in lines: 2 (which prevents Algorithm 3 from calling *Fall* infinitely many times in lines 3 and 8) and 13 of Algorithm 2 (which prevents it from calling *Fall* infinitely many times in lines 22 and 41). Moreover, each call of the *Fall* method consists of a finite number of steps since all lists: $A_o$, $B_o$, $B_o^+$ and $C_{o_i}$ are finite (again, by the fact that there is only a finite number of objects in a scene). The same applies to the *Check* method. It can only be called finitely often since all lists $A_o$ that *Check* relies on are finite and since there are only finitely many objects in a scene, only a finite number of such lists can be created during operation of the method.

### B. Comparison with alternative methods

As described in more detail in Sect. II, Siskind's method of determining stability of a scene (see [15], [16]) allows us to judge whether in a given layout *the whole scene* remains stable. Thus, if we wanted to apply the method to analyze the stability of a scene after one of its constituting blocks is destroyed, we would need to check if an appropriate assignment of linear and angular velocity to each constituent line segment of the structure is possible. In such an approach, however, several difficulties occur. First, our vision module does not allow us to precisely identify each joint as belonging to a particular type. Second, Siskind's method only makes it possible to evaluate if the whole *grounded* structure remains stable after the shot. In order to know which particular block of the given structure will remain stable, we would have to distinguish different substructures of the main structure, evaluate their stability and on that basis determine the falling blocks. This, however, ignores the mutual impact of different substructures on each other and the way they are grounded in each other.

The approach introduced by Blum *et al.* (see [17]) offers a sophisticated, dynamic analysis of the structure in question. Resting on estimated mass of objects, angles between them and friction coefficients, it calculates forces and tensions between blocks and returns an information if such a structure is stable. Again, if this approach is to be introduced into the AB shot scenario, it would face similar problems. It determines the (in)stability of the whole structure, whereas we want to know exactly which objects will fall in order to calculate a possible impact of a targeted block on other blocks.

The comparison of determining stability by the ERA agent (see [14]) and VI algorithm deserves a more extensive treatment. We will discuss two types of object configurations that are properly evaluated by our method, but not by the ERA agent.

In the first case we consider objects that seem to be not stable if we only consider the objects below them but turn out to be stable when objects above them are taken into account.

The ERA agent does not consider what is happening above an object whose stability is being determined (see [14]), therefore it would not be able to verify that $o_5$ from Fig. 8 is stable as long as $o_6$ and $o_8$ remain stable - in this configuration what is essential for $o_6$'s stability is that $o_5$ and $o_8$ press it from above. Our algorithm determines the centre of mass of objects lying above a considered object in order to check if the structure above makes it stable or is not sufficient to stabilize it (see Section IV-A). As a result, as long as we do not remove $o_1$, it recognizes $o_5$, $o_6$ and $o_8$ as stable objects.

Secondly, consider structures made of slanting objects lying one on another as in Fig. 9. Here $o_2$ is the only unstable object. In such a case the ERA agent uses a rule stating that if a slanting object is in surface-to-surface contact with another, already stable, slanting object and moreover the major part of the first object's vertical projection overlaps with the second object's vertical projection, then the considered object is stable (for a precise rule description see Rule 1.6 in [14]). As a result, the ERA agent would incorrectly recognize $o_2$ as stable, whereas VI recognizes $o_2$ as unstable since its centre of mass is to the left of the left-most connection point between $o_2$ and $o_1$.

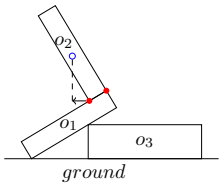



Fig. 9: An example of object configuration, in which ERA incorrectly recognizes $o_2$ as stable.

### C. Horizontal Impact

The second method, i.e., *Horizontal Impact*, enables us to estimate how the impulse of a bird directly hitting an object is propagated to other objects in the structure. We assume that given an object $o$ such an impact is propagated from $o$ to all objects that are $on\_right$, $lies\_on$ or $lies\_under$ $o$. If the impact is propagated to pigs or their shelters, they may fall down or be damaged which obviously is highly desired. The method assigns each object a numerical value of the so-called *force* that acts on that object. We use spatial networks introduced in Sect. III-B in order to check how the impact is propagated. Note that HI only investigates propagation of forces within the connected part of the spatial network the object $o$ is located in. It means that at the current stage of development of the algorithm quite frequently occurring cases of vertical blocks toppling on other objects (thus propagating force) are not covered by HI. The more detailed description of the method is as follows.

**First,** we take as an input a particular object $o$ we intend to directly fire at and the trajectory of the shot $t$.

**Second,** we initialize an array $forces$ of size $card(O)$ of forces acting on objects in the scene. Initially, $forces[o_i] \leftarrow 0$, for all $o_i \in O$.

**Third,** in order to estimate the force $forces[o]$ acting on the direct target $o$ of the shot, we need to take into account the type of the bird, the type of the target and the type of the shot trajectory. Since the impact of a bird on an object depends both on the object type and on the bird type, we introduce a $correlation$ function of these two as arguments which returns a value from the interval $[0, 1]$. The greater the value is, the higher the impact of the bird on the object. In Tab. I we present experimentally established values of the $correlation$ function.

TABLE I: Experimentally established $correlation$ function.

| $bird\_type$ \ $object\_type$ | $pig$ | $ice$ | $wood$ | $stone$ |
|---|---|---|---|---|
| $red$ | 1 | 0.8 | 0.65 | 0.3 |
| $blue$ | 1 | 1 | 0.5 | 0.25 |
| $yellow$ | 1 | 0.6 | 1 | 0.2 |
| $white$ | 1 | 0.7 | 0.8 | 0.6 |
| $black$ | 1 | 1 | 1 | 1 |

We assign to each type of trajectory (low and high) a value by $traj\_type : T \times O \rightarrow [0, 1]$ described in Def. 8.

**Definition 8** ($traj\_type$ function)**.** For a trajectory $t \in T$ and an object $o \in O$

$$traj\_type(t, o) = \begin{cases} 1 & \text{if } t = traj_{\rightarrow}(o) \\ 0.7 & \text{if } t = traj_{\nearrow}(o) \end{cases}$$

Then $forces[o]$, which in this case is the force acting on a directly hit object $o$ with a bird $b$ and trajectory $t$, is assigned a value in the following way:

$$forces[o] \leftarrow correlation\big(bird\_type(b), object\_type(o)\big) \cdot traj\_type(t, o)$$

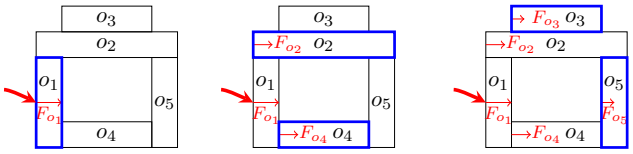
**Fourth,** we determine a set $P$ of objects that propagate the force in the current step and a set $N_P$ of their neighbours, i.e., objects that directly $lies\_on$, $lies\_under$ or $on\_right$ of some object from $P$. In Fig. 10(a) in the first step of the algorithm $P = \{o_1\}$ (marked blue) and $N_P = \{o_2, o_4\}$.

**Fifth,** for each element $o_i$ of $N_P$ we calculate the value of force propagated to $o_i$ from elements of $P$. If it is greater than previously assigned value to $forces[o_i]$, then new $forces[o_i]$ becomes the propagated force value. The propagated force depends on the stability of $o_i$ and of the maximum of values of forces that have been already propagated to objects $o_j \in P$ such that they either lie on or under $o_i$ or are located on the left of $o_i$. To capture it more formally, let $AO(o) = \{o' \mid lies\_on(o', o) \vee lies\_under(o', o) \vee on\_left(o', o)\}$. Let $forces\_max[AO(o)]$ be the maximal value of forces already propagated to elements of $AO(o)$. We set:

$$forces[o_i] \leftarrow \max\left(forces[o_i], \frac{forces\_max[AO(o_i)]}{1 + I\text{-}stability(o_i)}\right).$$

Additionally, we construct a set $N'_P \subseteq N_P$ as a set of objects to which new $force$ value has been assigned in the current step of the algorithm. In Fig. 10(a) in the first step of the algorithm force is propagated to $o_2, o_4$, therefore $N'_P = \{o_2, o_4\}$ in this step.

**Sixth,** until $N'_P \neq \emptyset$ we repeat steps **Fourth** and **Fifth** with a proviso that in each step the set $N'_P$ from the preceding step, i.e., the set of objects forces have just been propagated to, becomes the set $P$, i.e., the set of objects that propagate the force in the current step. New set $N_P$ is calculated as described in step **Fourth** and $N'_P$ becomes an empty set. In Fig. 10




$P = \{o_1\}$ $N_P = \{o_2, o_4\}$ $N'_P = \{o_2, o_4\}$ (a)

$P = \{o_2, o_4\}$ $N_P = \{o_1, o_3, o_5\}$ $N'_P = \{o_3, o_5\}$ (b)

$P = \{o_3, o_5\}$ $N_P = \{o_2, o_4\}$ $N'_P = \emptyset$ (c)

Fig. 10: The subfigures represent 3 steps of the HI algorithm. In each step objects from $P$ are marked blue, whereas $N_P$ and $N'_P$ are listed below a corresponding subfigure.

algorithm does 3 steps presented in Fig. 10(a-c) respectively and terminates.

Since there is only a finite number of objects in the scene and the propagated force from object $o$ is always lower than force propagated to object $o$, the algorithm always terminates.

**Algorithm 4** HORIZONTALIMPACT$(o, b, t)$

**Input:** an object $o$, a bird $b$, a trajectory $t$
**Output:** an array $forces$ of values of forces propagated to objects after $o$ gets hit by the bird $b$ with the trajectory $t$

initialize empty lists of objects $P$, $N_P$ and $N'_P$;
initialize an array $forces$ of doubles of size $card(O)$;
Set $forces[o_i]$ such that $\forall o_i \in O\ forces[o_i] \leftarrow 0$ ;
Set $forces[o] \leftarrow correlation(bird\_type(b), object\_type(o)) \cdot traj\_type(t, o)$; // calculate the direct force acting on $o$
Set $P \leftarrow \{o\}$;
Set $N_P \leftarrow \{o_i | lies\_on(o_i, o)$ **or** $lies\_under(o_i, o)$ **or** $on\_right(o_i, o)\}$;
**while** $N'_P \neq \emptyset$ **do**
  Set $N'_P \leftarrow \{\ \}$; // $N'_P$ becomes an empty list
  **for each** object $o_i \in N_P$ **do**
    Set a list of objects $AO \leftarrow \{o | lies\_on(o, o_i) \vee lies\_under(o, o_i) \vee on\_right(o, o_i)\}$;
    Set $forces\_max \leftarrow \max_{o_j \in AO} forces[o_j]$;
    **if** $forces[o_i] < \frac{forces\_max}{1+I\text{-}stability(o_i)}$ **then**
      Set $forces[o_i] \leftarrow \frac{forces\_max}{1+I\text{-}stability(o_i)}$;
      Set $N'_P \leftarrow N'_P \cup \{o_i\}$; // add $o_i$ to $N'_P$
    **end if**
  **end for**
  Set $P \leftarrow N'_P$; // new $P$
  Set $N_P \leftarrow \{o_i | lies\_on(o_i, o_j)$ **or** $lies\_under(o_i, o_j)$ **or** $on\_right(o_i, o_j)$ for some $o_j \in P\}$; // new $N_P$
**end while**

### D. Value estimation

Separately, VI and HI do not provide adequate reasoning. However, the combination of the abovementioned methods enables us to predict objects' behaviour even in complex structures. For an example see the structure depicted in Fig. 11, where hitting $o_1$ makes $o_2$, $o_3$ and $o_6$ (a pig) fall down because they lost one of their supports. On the other hand, the impact of a shot propagates to $o_7$ (another pig) which gets battered. In order to reason correctly about this situation VI and HI need to be used simultaneously.

After performing VI and HI methods, we are able to define a function $influence : O \times O \times B \times T \rightarrow [0, 1]$ which assigns a value of the overall influence that an object hit has on another object if the shot is performed with a given bird and trajectory. Values of the $influence$ function belong to the interval $[0, 1]$, where $influence(o_1, o_2, b, t) = 0$ means that if we fire at $o_1$ with a bird $b$ and trajectory $t$, $o_1$ will have no influence on $o_2$, whereas $influence(o_1, o_2, b, t) = 1$ means that if $o_1$ falls after a direct shot at it, then $o_2$ automatically falls, too. Formally, we introduce the function in the following definition.

**Definition 9** ($influence$ function)**.** For objects $o_1, o_2 \in O$, a bird $b \in B$ and a trajectory $t \in T$

$$influence(o_1, o_2, b, t) = \begin{cases} 1 & \text{if } o_2 \in fall \\ forces[o_2] & \text{otherwise} \end{cases},$$

where $o_1$ is a directly hit object, and the lists $fall$ and $forces$ are obtained by running VI and HI respectively.

In order to choose the best shot in a given AB scene, we need to estimate the overall value of available shots. This value depends on the type of a bird to be fired, the type of a shot trajectory (high or low parabola) and the block which is a direct target of the shot. More precisely, the value of the shot, i.e., the value of the function $value : B \times T \times O \rightarrow \mathbb{R}$ increases whenever the I-stability of a direct target decreases or its influence on interesting objects increases, i.e., pigs or their shelters. The exact description of the $value$ function is provided in the forthcoming definition.

**Definition 10** ($value$ function)**.** For a bird $b \in B$, a trajectory $t \in T$ and an object $o \in O$

$$value(b, t, o) = \frac{1}{I\text{-}stability(o)} \cdot \sum_{o_i \in O} ivalue(o_i) \cdot influence(o, o_i, b, t),$$

where $ivalue : O \rightarrow \mathbb{R}$ maps an object into its interesting value, i.e., for an object $o \in O$:

$$ivalue(o) = \begin{cases} 100 & \textit{if } o \text{ is a pig} \\ shelter\_val(o) & \text{otherwise} \end{cases}.$$

Since the value of $shelter\_val$ is always less than 100, then $ivalue$ function has a greater value for a pig than for any other object.

What we call "the best shot" is the one with the maximal $value$, which indicates its highest possible influence on pigs and their shelters. While the goal of the game in each level is to kill all pigs, the strategy of searching for a shot that will affect pigs and their shelters to the highest degree seems reasonable. Its empirical evaluation is presented in the following section.

## V. EVALUATION AND DISCUSSION

The first major proving ground for our program was the 2014 Angry Birds AI Competition [21]. The Angry Birds AI Competition is an annual event where the participants' task is to develop computer programs that play AB without human intervention. These programs compete on pre-designed AB levels (not previously known to the competitors). As the organizers claim [21], the long-term goal of the competition is to build an AI agent that can play new AB levels better than humans. We finished 5th in that competition.

Since then we have upgraded our program and compared the new version with the benchmarks with the results of agents

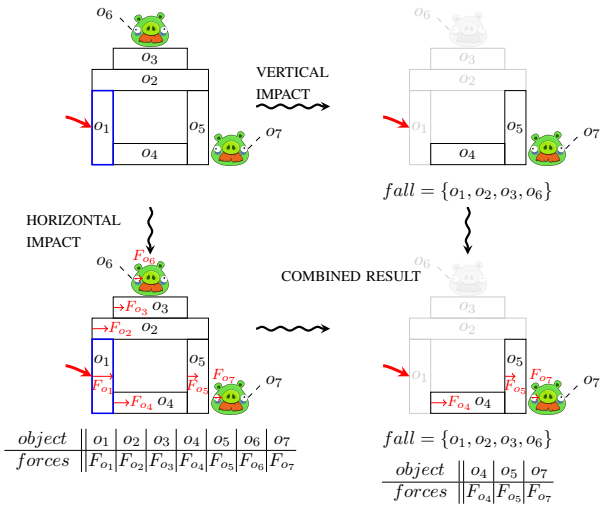



Fig. 11: Combination of VI and HI.

which participated in 2014 as well as 2013 edition of the competition. The benchmarks are available online (aibirds.org), thus in Tab. II we only present the scores of the best agent (PlanA+), our score, the score of agent using ERA (QSR approach) and the score of Naive Agent. PlanA+ introduces the resistance factor for each pair consisting of a bird type and a block type (similar to *correlation* considered in Tab. I). On that basis, PlanA+ for each shot aimed at a pig or an exploding block (TNT) counts the number of objects predicted to be destroyed and chooses the best shot. The approach is simple but due to a long optimisation of the parameters (55 hours of parallel computations on 20 computers) it obtained very good results in the competition. Naive Agent is a program which randomly selects a pig from a scene and targets its shots directly at it, even if it is not reachable, i.e., even when it is effectively sheltered by other blocks. Apart from selecting a pig and a trajectory mode - low or high - Naive Agent does not perform any reasoning. As Tab. II shows, our agent obtained better overall results than ERA and Naive Agent. Furthermore, our agent outplayed the best agent (PlanA+) on 8 out of 21 and had the best score of all 30 agents on 2 levels.

A separate issue is the prediction quality of our *Vertical Impact* algorithm. In order to test it, we have conducted a series of experiments. On each of the 21 levels of the Poached Eggs scenario we have chosen blocks that are reachable in the initial situation. For each such a block $o$ we have compared the *Vertical Impact* prediction about stability of the structure without $o$ with the actual situation that occurred after destroying $o$ in the game. More precisely, we denote by $P$ a set of blocks that *Vertical Impact* predicted to fall/be destroyed and by $F \subseteq P$ a set of blocks that have been predicted to fall/be destroyed *and* in fact have fallen/been destroyed in the Angry Birds game. The ratio $R = \frac{|F|}{|P|}$ is 1 if all of the predicted blocks have fallen/been destroyed and is 0 if none of the predicted blocks have fallen/been destroyed. If the value of $R$ is close to 1 in all the tests, the algorithm is *sound*, i.e., it does not predict a block to fall if it does not actually fall.

The result of the performed tests are as follows. We have

TABLE II: Benchmarks of AB programs. When our agent has the highest score on a given level, it is written in bold.

| Level | PlanA+ | Our agent | ERA agent | Naive Agent |
|---|---|---|---|---|
| 1-1 | 30480 | **32880** | 29210 | 29760 |
| 1-2 | **62370** | 43350 | 42760 | 43250 |
| 1-3 | 40620 | 40350 | **41610** | 40180 |
| 1-4 | **29000** | 19090 | 27990 | 10590 |
| 1-5 | **69440** | 65160 | 63840 | 62490 |
| 1-6 | **36970** | 35270 | 25700 | 14980 |
| 1-7 | 32020 | 37420 | **45990** | 22150 |
| 1-8 | 47320 | **48180** | 23390 | 35840 |
| 1-9 | 26440 | 42310 | **45930** | 36050 |
| 1-10 | **56830** | 55560 | 49570 | 52570 |
| 1-11 | 47240 | **49440** | 38570 | 39310 |
| 1-12 | 58210 | 58190 | **54990** | 48660 |
| 1-13 | 34010 | **39040** | 32270 | 30000 |
| 1-14 | **65640** | **65640** | 57550 | 45640 |
| 1-15 | **54910** | 31450 | 47280 | 44190 |
| 1-16 | 57530 | 53280 | **63000** | 52300 |
| 1-17 | **51190** | 42090 | 42770 | 39530 |
| 1-18 | 52120 | **56080** | 48290 | 39590 |
| 1-19 | **39440** | 32210 | 22040 | 29460 |
| 1-20 | **45980** | 38410 | 36910 | 40140 |
| 1-21 | 64620 | **70130** | 53710 | 0 |
| Total | 1002380 | 955530 | 893370 | 756680 |

conducted 60 experiments. In one experiment (presented in Fig. 12) *Vertical Impact* predicted 3 blocks to fall while in fact only one of them did, therefore in this case $R = \frac{1}{3}$. In the remaining 59 experiments $R = 1$. We conclude that the algorithm is *sound* and in what follows we discuss situations in which our algorithm failed to predict the correct number of falling blocks and those in which it succeeded when other methods would not.

Fig. 12: Left: initial sitiuation (red arrow symbolizes a shot trajectory, $o_1$, $o_2$ and $o_3$ were predicted to fall by VI); middle: $o_2$ and $o_3$ remain stable despite being predicted to fall by the algorithm; right: the test was repeated and $o_2$ and $o_3$ fell.

We consider two specific examples of application of the algorithm. The first situation (Fig. 12) is the only instance, where blocks predicted to fall by our algorithm remained stable. These blocks remained stable due to the horizontal impetus that moved them to the right and made them stuck. This example shows that there are some unpredictable events that may go against our algorithm's predictions. These however occur very rarely. In particular, a repeated test on that level resulted in the scenario predicted by our algorithm.

The situation shown in Fig. 13 proves the strength of our algorithm. A block that was supporting other blocks in the structure has been removed, yet the whole structure remained stable. In particular, in Fig. 13 one of the highlighted blocks lost its support and if considered without its surrounding structure, it would be predicted to fall, however due to the

pressure of the blocks above it, it remains stable. This kind of reasoning is novel, in particular it is not present in the method described in [14].

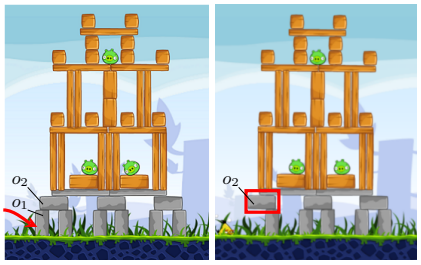



Fig. 13: Left: initial situation (red arrow symbolizes a shot trajectory); right: $o_2$ remains stable due to the pressure of the blocks above.

## VI. Conclusions

In this paper, we presented a program autonomously playing AB, whose main aim was to successfully complete subsequent game levels. Although the approach is dedicated to playing AB, the introduced method may be used to perform reasoning in other applications that consider two-dimensional physical space. The main strong points of the method are:

- The program involves qualitative representation of a two-dimensional physical environment that tries to employ similar (qualitative) categories people use when reasoning about space.
- The reasoning methods introduced to the algorithm, namely *Vertical Impact* and *Horizontal Impact*, are a successful mix of qualitative and quantitative reasoning methods.
- Practical performance of the program which resulted in a high total score gathered in all 21 levels of AB Poached Eggs scenario and in a higher (or equal) score than the best agent participating in 2014 AB AI Competition (Plan A+) in 9 levels.

On the other hand the approach has its limitations:

- In the approach only shooting at the left top point of an object is considered. Although it gives good practical results, in some situations shooting at other points would be better.
- The combined reasoning performed by VI and HI does not take into account the effects of rolling balls or force propagation between non-adjacent objects (e.g., when a high object is toppling onto another one).
- The approach does not involve planning, i.e., only the best shot in a given game scene is considered.

We see the evolution of the algorithm threefold. Currently the algorithm always picks a unique block with the highest value as a target of the shot. In the modified version we want it to distinguish, say, 20% of blocks with highest values and pseudo-randomly choose between them assuming that the chance of a block being selected is proportional to its value. Second, we would like to introduce machine learning methods to the program. Values of coefficients standing by constituents of final value estimation, e.g., $I\text{-}stability$, $shelter\_value$, etc., could presumably be modified so that the scores of the program improve. We plan to employ techniques of evolutionary algorithms to make the program amend the coefficients according to the results obtained. Finally, we want to fully integrate our two algorithms, namely VI and HI. We would like HI to take over the role of the basis for the whole integrated method. VI will only be run for a given object provided that the force value assigned to this object by HI exceeds a certain threshold value representing the minimal value of force acting on a block sufficient for destroying or overturning it.

## Acknowledgments

The authors would like to express their gratitude to the anonymous referees whose detailed and insightful comments help significantly improve this paper. This paper is a result of the project funded by the National Science Centre, Poland (grant number: DEC-2011/02/A/HS1/00395).

## References

[1] H. Gardner, *Frames of Mind: The Theory of Multiple Intelligences*. New York, NY, USA: Basic Books, 1983.
[2] J.-H. Shin and H.-B. Jun, "A study on smart parking guidance algorithm," *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 299 – 317, 2014.
[3] P. Zips, M. Böck, and A. Kugi, "A fast motion planning algorithm for car parking based on static optimization," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Tokyo, Japan, Nov. 2013, pp. 2392–2397.
[4] T. Hirst, "Robot cars, part 1: Parking the future for now," May 2013. [Online]. Available: http://www.open.edu/openlearn/science-maths-technology/engineering-and-technology/technology/robot-cars-part-1-parking-the-future-now
[5] M. A. Greenspan, J. Lam, W. Leckie, M. Godard, I. Zaidi, K. Anderson, D. C. Dupuis, and S. Jordan, "Toward a competitive pool playing robot: Is computational intelligence needed to play robotic pool?" in *CIG*. IEEE, 2007, pp. 380–388.
[6] B. Kuipers, "Qualitative simulation," *Artificial Intelligence*, vol. 29, no. 3, pp. 289–338, 1986.
[7] K. D. Forbus, "Readings in qualitative reasoning about physical systems," D. S. Weld and J. d. Kleer, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Qualitative Physics: Past Present and Future, pp. 11–39.
[8] J. de Kleer and J. S. Brown, "A framework for qualitative physics," in *Proc. 6th Ann. Conference of the Cognitive Science Society*, 1984, pp. 11–18.
[9] A. G. Cohn and J. Renz, "Qualitative spatial representation and reasoning," *Handbook of knowledge representation*, vol. 3, pp. 551–596, 2008.
[10] D. A. Randell, Z. Cui, and A. G. Cohn, "A spatial logic based on regions and connection." *KR*, vol. 92, pp. 165–176, 1992.
[11] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.
[12] P. Balbiani, J.-F. Condotta, and L. F. del Cerro, "A new tractable subclass of the rectangle algebra," in *IJCAI*, vol. 99, 1999, pp. 442–447.
[13] P. Zhang and J. Renz, "Qualitative spatial representation and reasoning in Angry Birds: First results," in *27th International Workshop on Qualitative Reasoning*, 2013, p. 123.
[14] ——, "Qualitative spatial representation and reasoning in Angry Birds: The extended rectangle algebra," 2014.
[15] J. M. Siskind, "Visual event classification via force dynamics," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence*, Austin, TX, USA, 2000, pp. 149–155.
[16] ——, "Method of determining the stability of two dimensional polygonal scenes," *United States Patent*, no. US 6,693,630 B1, pp. 1–19, 2004.
[17] M. Blum, A. Griffith, and B. Neumann, "A stability test for configurations of blocks," *Artificial Intelligence Memo*, no. 188, 1970.
[18] P. E. Nielsen, "A qualitative approach to rigid body mechanics," Ph.D. dissertation, Champaign, IL, USA, 1988, aAI8908788.
[19] P. Pu, "Simulating both dynamic and kinematic behaviors of mechanisms," in *Proceedings of the Third Workshop on Qualitative Reasoning*, Stanford, CA, USA, 1989, pp. 1–13.
[20] C. H. Papadimitriou, *Computational complexity*. John Wiley and Sons Ltd., 2003.
[21] J. Renz, "AIBIRDS: The Angry Birds Artificial Intelligence Competition." in *Proceedings of the 29th AAAI Conference*, 2015, to appear.