AngryHEX: an Artificial Player for Angry Birds Based on Declarative Knowledge Bases

UNIVERSITÀ DELLA CALABRIA

Dipartimento di Matematica e Informatica

TU WIEN

FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Francesco Calimeri[1] Michael Fink[2] Stefano Germano[1]
Giovambattista Ianni[1] Christoph Redl[2] Anton Wimmer[2]

[1] Dipartimento di Matematica e Informatica, Università della Calabria
[2] Technische Universität Wien, Institut für Informationssysteme

Angry Birds

A group of "Angry Birds“ seek for revenge on some green pigs (the "Piggies") that stole their eggs and ate them. Armed with a big slingshot, the Birds jump onto their enemies in order to… eliminate them!

Gameplay

- Each level includes a number of birds (each with different characteristics), a slingshot to launch them towards the targets, and one or more pigs enemies.
- By means of the bird launches, the player must be able to kill all the pigs present, hitting them directly or making some structures tumble upon them.
- The structures consist of "bricks" of different sizes and materials (such as ice, wood or stone) more or less fragile, and therefore more or less easy to demolish.

Angry Birds AI Competition

- Worldwide competition open to research groups able to develop an artificial intelligence that knows how to play Angry Birds
- The long-term goal is to get sooner or later an intelligent agent playing better than a human being, when both face levels that have never played before
- The last competition took place in Beijing, China, in conjunction with IJCAI 2013
- http://www.aibirds.org

AngryHEX - Motivation

- **Goal:** Design a declarative agent that plays the game
- **Challenge:** Plan “optimal” shots under consideration of some physics
- **Our means:** HEX-programs, i.e., Answer Set Programming (ASP) with external sources and other extensions.

Architecture of our Agent

- On top of the standard framework provided by AI Birds Competition Organizing Committee (browser plugin, vision module, etc.)…
- …Actual reasoning is represented by an HEX-program (a logic program), and reasoning is performed by system supporting HEX-programs that computes the desired target, given the information provided by the vision module
- Agent extracts the target from the models of the logic program

It relies on tactic and strategy:

- The tactic module aims at completing one level
- The strategy choose the next level to play

Tactic Layer

Tactics **declaratively** represented by the HEX-program:

- Consider each shootable target (objects reachable by a direct and unobstructed path from the slingshot)
- Compute the estimated damage on each other object if the given target is hit, directly or indirectly, taking into account different bird types
- Damage estimations follow causal chains combined with an estimate of the loss of kinetic energy
- Rank the found targets (=answer sets) using weak constraints: we add malus points for each possibly missed pig (…and much more)
- Learn from history: since best scores are saved, never play a level in the same way more than once: always looking for new ways of killing pigs!

HEX-Encoding for the Tactic Layer

- **Input:** Scene information extracted by the vision module is encoded as facts (positions, size and rotation of pigs, ice, wood and stone blocks, slingshot, etc.)
- **Output:** Answer Sets (Models) of the HEX-program contain a dedicated atom representing the “elected” target
- Results of the physics simulation can be accessed by the HEX - program via eternal atoms, e.g.:
  - decide if object ***B*** falls when ***A*** falls
  - decide which objects intersect with the trajectory of a bird after hitting a given object
  - compute distances between objects
  - etc…

Strategy Layer

- The strategy module decides the next level to play
  1. First, try to play each level once
  2. Then, chose the level where the agent performed worst with respect to the best scores
  3. Then, play levels where the agent performs better than the best scores, but the difference is minimum to the best scores

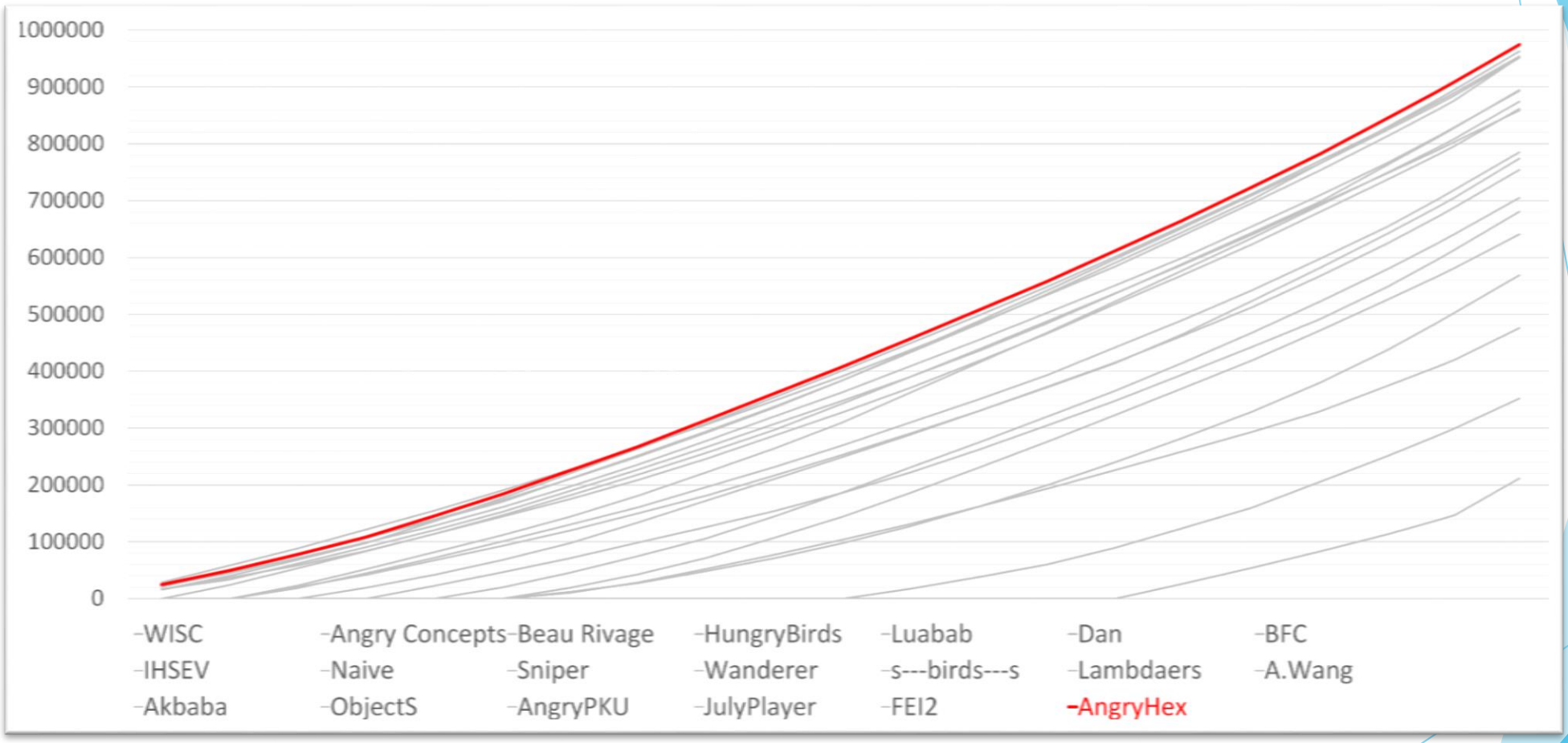
Logic rules: example

$$pushDamage(Obj_B,P_A,P) \leftarrow pushDamage(Obj_A,\_,P_A),$$
$$P_A > 0,\ \&canpush[ngobject](Obj_A,Obj_B),$$
$$pushability(Obj_B,P_B),\ P = P_A * P_B/100.$$

2nd AI Birds Competition

- Venue: Beijing, China, in conjunction with IJCAI 2013
  - 20 teams
  - 12 Countries
  - 5 Continents
- Results:
  - Qualifications: 1st (out of 20)
  - Quarter of finals: 1st (out of 8)
  - Final overall rank: 4th

Results



Cactus plot of the cumulative score for the teams participating to the Competition ("Poached Eggs" Level set).

Future works

- Combine objects which behave like a single one
- Plan over multiple shots
- Realize strategies declaratively (currently implemented in Java)
- …any ideas? ;)

Thank you for your attention