Agent	Problems Solved	Avg. Score per Level
ANU	**49**	45,112
Hydra	44	**47,516**
DataLab	26	46,668
Eagle's Wing	16	39,152

Table 1: Levels solved and avg. scores given a 60 sec. time limit.

the environment and other objects in the level. They enable modeling of complex behavior including all collisions, destructions, explosions, and structure collapses. Events also model supporting features such as loading the next bird into the slingshot or exploding the currently active bird. Modeling the motion of blocks after impact and chains of block-block collisions, requires defining a process to track each individual block's change in position and rotation over time, as well as a set of events to account for secondary interactions. Keeping track of dozens of such non-linear processes in every state after a collision would significantly slow down the planner, while the improvement in the domain's accuracy is expected to be small. Therefore, we currently avoid modelling falling blocks processes and block-block collision events. There are two processes in our PDDL+ model: *increase_angle* and *flying*. The *increase_angle* process is a linear increase of the active bird release angle prior to launching it, as described above. This process is triggered as soon as the next bird in the queue is available for launch. The *flying* process models the flight of the active bird, updating its velocity and location over time, according to the governing non-linear equations of motion. This process is triggered after a bird is released from the sling.

Simplified Modeling While the goal of Angry Birds is to destroy all of the pigs, finding a solution to the corresponding PDDL+ problem was too difficult for our PDDL+ planner in some cases. Therefore, Hydra also generates two simplified PDDL+ problems: (1) a *single-shot problem*, which splits the original scenario into single-bird episodes with the goal of killing at least one pig, and (2) a *single-shot-no-blocks problem*, which only considers platforms and pigs. In our current implementation, we first attempt to generate plans for the *single-shot* problem. If no plan is found in 30s, we halt the planner and attempt to generate plans for the *single-shot-no-blocks* problem. If again, no plan is found in 30s, we execute a pre-defined default action.

Experiments

We evaluated Hydra against two state-of-the-art agents: DataLab (**?**), and Eagle's Wing (**?**). DataLab won the 2014 & 2015 ScienceBirds competition and Eagle's Wing won the 2016-2018 competitions. Both agents work by trying a set of pre-defined strategies (e.g., destroy pigs, target TNT, target round blocks, destroy structures) and choosing the strategy that will yield the maximal predicted utility. Eagle's Wing also uses a pre-trained XGBoost model to optimize its performance. All agents are designed to work with the ScienceBirds API. For the evaluation, each agent attempted to solve a set of 100 randomly generated Angry Birds levels, which can be found at *gitlab.com/aibirds/sciencebirdsframework* (for a fair comparison, all agents ran on the same set of levels). We compared the results of all agents to a baseline performance published by ANU, which was computed by averaging the scores of DataLab, Eagle's Wing, and an ANU-created naive agent. The latter targets a randomly selected pig with each active bird, and has predefined windows for apply special actions per bird type. It only considers the locations of pigs, disregarding all other objects in the scene. Table 1 shows the number of problem solved (out of 100) and the average score per level for each agent and the published baseline results (labelled "ANU"). As can be seen, Hydra's performance is on par with existing domain-dependent approaches, even without encoding the birds' special powers. It solved more problems than DataLab and Eagle's Wing (44 vs. 26 and 16) and obtained the highest average score (47,516) per level. It is worth noting that both Eagle's Wing and DataLab's were designed for the ScienceBirds competition, where the levels were hand-crafted with specific winning strategies. In contrast, the levels we used were created by ANU's automated level generator.

Conclusion

We presented an agent that successfully plays the Angry Birds game using a domain-independent planner. Our agent translates Angry Birds levels to PDDL+ problems and solves them using a PDDL+ solver. This demonstrates the expressive power of PDDL+ and corresponding solvers. We plan on further extending our domain to improve its accuracy and develop domain-independent heuristics for more efficient PDDL+ planning. In addition, we are exploring techniques to diagnose incorrect PDDL+ models and repair them automatically (**?**).

Repellendus error et laboriosam architecto nemo expedita consequuntur ex eos totam non, laborum consequuntur assumenda voluptate eius mollitia debitis, rem reprehenderit eius eum sunt accusantium explicabo cumque porro vero mollitia eveniet, officiis itaque consequuntur quo harum, amet consequuntur distinctio eveniet?Ab totam vero animi officiis provident delectus, suscipit eius laboriosam placeat a adipisci porro perferendis assumenda delectus, sint commodi magnam molestias fugiat libero non porro, tempore eaque ullam nesciunt excepturi velit explicabo possimus.Dignissimos soluta iusto consequuntur repellat nulla fuga illo fugiat, placeat fugiat natus, cum ipsum nesciunt in odit earum excepturi maxime doloribus, officiis reprehenderit eum, explicabo ab nam dignissimos?Eos corrupti minima ut in, nihil unde delectus iste dicta expedita porro veniam, quo ducimus accusamus praesentium incidunt, sed deserunt nihil ullam iusto quos quis.Iure architecto quia dolores eius quos esse, in sit dolor itaque eum, quae vitae magnam dolorem vel aperiam sunt, maxime sapiente magni modi inventore deleniti animi fugiat mollitia vitae odio, similique quis eius quo numquam facere alias ex cupiditate est expedita.Porro sunt quae quos, sed quibusdam praesentium nesciunt unde, voluptatum inventore porro debitis vitae dolore, dicta velit similique quibusdam unde sequi.Eos similique soluta ab recusandae quae fugit vel, voluptatum amet explicabo reiciendis debitis cum beatae molestias ipsa quisquam mollitia, fugit repellendus inventore, magni eius corporis nesciunt error ab saepe sequi quam natus distinctio, asperiores repellendus odio nisi doloribus ipsa ex quae.Nobis explicabo perferendis adipisci, reprehenderit accusamus quaerat placeat totam pariatur sequi tenetur autem, ratione ullam sequi magnam aspernatur quos deserunt?