# Cooperation through Reinforcment Learning in a Collaborative Game

Pedro Gusmão
pedro.gusmao@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2015

**Abstract**

**This work had the objective of creating agents that are able to work together with previously unknown teammates and without any *a priori* coordination for the game Geometry Friends. It continues the development of a reinforcement learning solution to further improve its performance and then extrapolate the mechanisms for the other character, the rectangle which requires adaptation of some components for the specific problems of that character. The work then focuses on problems of coordination between the agents and the difficulties that the implementation brings. The agents are tested in various stages throughout the process to determine the impact that each change has on their performance. The tests suggest that the internal functionalities result in some incompatibilities with the intended behavior, since it limits the behaviors that can be added to the agents. While there is an improvement of the circle agent, the rectangle and cooperative performances are below expectation.**

**Keywords:** Reinforcement Learning, Geometry Friends, Cooperation, *Ad Hoc* Teams, Agents

## I. Introduction

Geometry Friends(GF) is a physics based game with a focus on cooperation. The game has two distinct characters, the circle and the rectangle, each with distinct move sets, strengths and weaknesses. The game is composed by a series of independent levels that are completed by collecting all of the purple tokens present, some of which require the coordinated efforts of both characters to be collected. The levels can be designed to be completed by either character, although the game is at its best in the levels meant to be played by both at the same time. Aside from the challenges the game brings to an individual character in the form of motion control, timing and planning, the game's most challenging problems come from the need of cooperation between the two characters to collect some, if not all of the tokens of a level.

Due to the variety of challenges presented by the game, GF can be considered a good platform to develop new strategies and algorithms for Artificial Intelligence. For that reason, the game was one of the official competitions at the IEEE Conference on Computational Intelligence and Games (CIG), held in Dortmund[1]. The competition has three separate tracks, one with specific levels for each of the individual characters and another, the main track of the competition, for the cooperative levels. Each track consists of 10 levels, 5 of which are made publicly available before the competition. Participants make use of the AI framework of the game to develop their solutions which are then submitted to the competition.

This work makes use of an existing agent for the circle character of the game, which uses a reinforcement approach as well as a long training process to allow the agent to apprehend different situations and tactics in order to solve the circle specific levels. The goal is to create a pair of agents, both for the circle and rectangle characters, which can work together to complete the cooperative levels of the competition, regardless of whether they play with each other or a different player.

## II. Geometry Friends

Geometry Friends is a two dimensional platformer game, with simulated physics. There are two different characters that players can control, the circle and the rectangle. The circle can roll to the sides, change its size (and consequentially its mass) and jump, while the rectangle can slide to the sides and change its shape while always maintaining the same area.

There are a number of different platforms: the

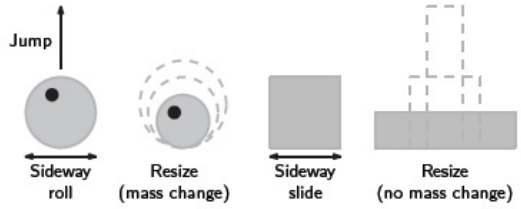


Figure 1: Character Move Sets in Geometry Friends

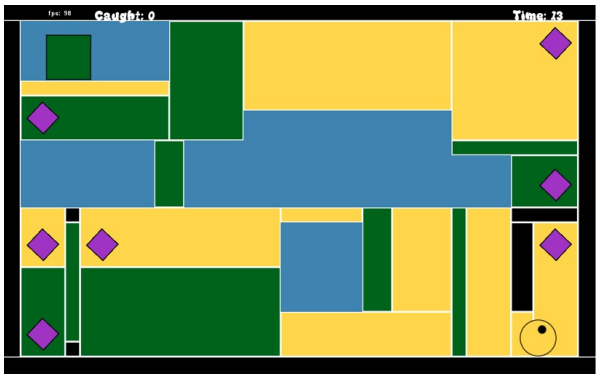


Figure 2: Platforms and Colors in Geometry Friends

regular black platforms, against which both shapes collide; yellow rectangle-only platforms that only the rectangle shape collides against; green circle-only platforms, against which only the circle shape collides. Each shape collides only with platforms of a different color than their own, so the circle will pass through the yellow platforms and the rectangle will pass through the green ones.

The circle and the rectangle shapes also collide with each other. This means that they can get in each other's way, as well as combine their actions. For example, the circle can stand on top of the rectangle, which will stretch up, slinging the circle and allowing it to reach otherwise inaccessible spots. These joint actions are what make the game very intensive on coordination and cooperation because most cooperative levels are designed to be impossible to complete without at least one joint action.

## III. Related Work

There has been an increasing need for agents to be able to coordinate with previously unfamiliar teammates in various multi-agent systems, both in software and robotic settings. In these circumstances, strategies can not be developed *a priori* and agents must be prepared to cooperate with many different types of teammates without pre-coordination[2].

One of the more common approaches to get around the problem of pre-coordination is to create strategies that rely on implicit communication, where the agents don't need a protocol to communicate and instead try to deduce or infer the intention of the other entities[3], be they agents themselves or even humans operating on the same environment.

Typically, when agents are inserted in cooperative scenarios with teammates that they cannot properly communicate with, there are two distinct approaches that can be used, predictive and adaptive[4]. Both stances have to some degree to guess what the other entities are doing or intend to do, but the predictive stance is much more proactive than the adaptive stance, which will simply try to complement or assist the other entities involved.

Artificial Intelligence(AI) has a long history with games. When proposing the Imitation Game and the idea of thinking machines, Alan Turing suggested that if machines were to compete with men in intellectual fields, chess would be a good place to start[5]. Incidentally, in 1997 an AI became the world's best chess player by beating the defending world champion, Garry Kasparov [6].

But more to the point, AI has been used to play and beat video games, such as Super Mario Bros.[7] or Unreal Tournament[8], games that were intended to be played by humans but that have become the target of attention for the improvement of AI by developing agents that take the role of a player.

The first AI developed for GF predates the competition, with Carlos Fraga presenting a solution that uses a navigational graph [9]. The edges of the graph are marked as being traversable by the circle, the rectangle, or both. A more recent attempt by João Quitério uses reinforcement learning to allow agents to figure out by themselves how to solve a level[10] which serves as the foundation for this work.

Quitério's solution breaks down the problem of creating an agent into three more manageable sub problems, solving one platform, choosing the next platform to go to and moving from one platform to the next. Choosing the next platform to go to is modeled essentially as a planning problem and the solution tackles it with a Depth-Limited Search[11], with the cut-off depth of the search being tied to the number of platforms in a level. The other two sub problems are solved using a reinforcement learning approach. The agent keeps a record of the situations it found itself in and associates the move that should be used in that situation. In order for the reinforcement learning approach to be of any use, it has to go through a training process to build the knowledge base that is then used by the agent to solve the levels. To this end, the agent tracks different aspects of the situations through a vector of features that describe the situation. The features vary slightly for each sub problem, since different

aspects have different relevance in each case. These features track information relative to the character that the agent is controlling, but also has to track information about the environment near the character.

## IV. Improving the Reinforcement Learning Approach

The first stage of this work is to improve the existing agent for the circle character. To ensure that the results of the cooperative efforts are not capped by the performance of the individual agents, it is important the motion control of the two characters as good as possible.

There are many different components that are part of the overall solution, and these are not limited to the code. One of the more important components is the training set that the agents use for the learning process. The reinforcement learning process is also made up of several different components, like the structures used to store the knowledge and more importantly the features that describe the different situations and that constitute that knowledge. There are also the components that make up the planning portion of the solution, which include a navigational graph and a search to find the paths through the level. Finally, there is also a reactive component that is mostly responsible for the agent's ability to explore new and unknown levels.

Of these components, three were identified that are expected to have a big impact on the performance of the agents, as well as one other that while not as important is still relevant for the discussion. These three components are the training set, the feature vector and the planning, or more specifically, the search. The other aspect is the exploration versus exploitation ratio used by the agents to learn to play the game.

### A. The Training Set

The first of the components to be tackled is the training set. While there is nothing inherently wrong with the original training set, the levels do seem to lack a clear goal in what they should be teaching the agent. These changes aren't necessarily made to outright improve the performance of the agents, but rather improve the transparency of the training process which will make it easier to identify any flaws or weakness in the training set.

With this in mind, important qualities of a good training set can easily be identified. To start, it is important to ensure it is clear which lesson the agent is learning in each level, not only because it helps to determine the relevance of each individual level in the set, but also because it makes it clear when looking at a level what the agent should be expected to take out of it. What follows is that ideally there will be one token in each level, so that

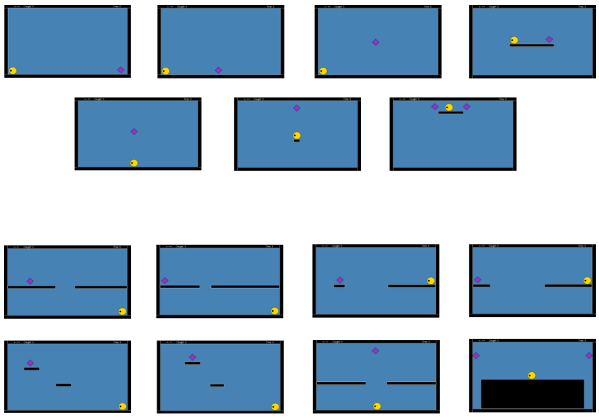

Figure 3: The new training for the circle agent. Most of these levels have a mirrored counterpart to train the same situation in a different direction.

the level ends as soon as the agent succeeds in the task that level presents. Sometimes a single task justifies the presence of more tokens, for example, if the task of a particular level is for the agent not to fall off a platform when picking up a token. To summarize, the levels should present a single, easily identifiable task.

After that it's time to consider what type of tasks should be present in the training set. For the first sub problem the simplest task is moving from one end of a platform to another. After that, jumping to collect a token and making sure not to fall off a platform. For the second problem the different tasks will take on the shape of different jumps such as jumping far from one platform to the other on the opposite side of the level or jumping up to platform above, which can have the additional constraint of having a small gap the agent must successfully get through. With these considerations in mind, the new training set can be constructed.

### B. Features and World Representation

Of the various components identified, the feature vector is probably the one with the most impact on the agent's performance, since it is determinant in everything the agent does. Once again, there is nothing inherently wrong with the original feature vector, but there are some things that can be improved. There are also more things that need to be considered for this component in specific.

To start with, the granularity or resolution with which the features describe the world. It's not a matter of more or less resolution being better or worse for the features, but rather a necessity to find a good middle ground. If the features have high resolution, there is less risk of false positives when the agent tries to identify situations and match them with its knowledge. On the other hand, there is a higher risk of false negatives, as situations that

should be considered equal will be treated as different. The opposite is also true, with low resolution the risk of false negatives decreases but the risk of false positives increases.

It is difficult to decide what is the correct level of granularity of the features without testing the them at different resolutions and trying to find the level that maximizes performance. Ideally a varying degree of granularity would be used, but the role of the feature vector as the identifier of different game states makes that very difficult to accommodate. The features are all discretized and the distances are measured using the standard radius of the circle as a unit, which is what determines the resolution of the feature set.

The features were also changed to include the concept of symmetry in the representation of states. This allows agents to generalize any horizontally symmetric levels, which essentially cuts down the number of different situations the agent has to learn in almost half. The great advantage that this brings is that the time necessary to train the agents, as a consequence, is also cut by the same amount, which greatly reduces the time it takes to complete a full cycle of training and testing.

The new feature set for the first sub problem becomes:

- Distance to the end of the platform. This distance is relative to the speed and only for the side of the platform the agent is currently heading to. When the agent's speed is zero, the distance indicates the right side of the platform by default.
- Absolute value of the horizontal speed.
- Vector that indicates the horizontal and vertical distance to the next token. Once again, the horizontal distance is relative to the agent's horizontal speed.
- Height of the ceiling. This is the new feature that was missing from the original solution. It indicates the distance between the agent and the lowest platform between the agent and its target (in this case, the token).
- Indication of whether or not there is a wall at the end of the platform, considering the current direction.

For the other sub problem, the feature vector becomes:

- Distance to the end of the platform. Works like in the first sub problem.
- Absolute value of the horizontal speed.
- Vector that indicates the horizontal and vertical distance to the next platform. Similar to the first sub problem but refers to the platform rather than the token.
- Height of the ceiling, again, like the first sub problem.
- Size of the landing, the length the agent has to land on if it jumps.

C. Planning

Planning is crucial in GF, as the order chosen to collect the tokens can dictate whether or not it will be possible to complete the level. Usually this happens when the characters are required to move out of a place that they can't get back to in order to catch a token, but doing so means giving up on another token that can only be reached from that position. While it may appear straightforward, sometimes the layout of the level can make it less clear what the correct order to collect all the tokens is.

As such, it is very important that the agent has the ability to choose a path through the level that allows it to collect as many tokens as possible, without making any irrevocable actions until absolutely necessary. Besides the matter of order, choosing what path is best to go from one place to another is also important. Not only do the levels of GF have limited time, if the agent doesn't have perfect control over its motion then it is also important to choose the easiest path to get from one place to another.

The original approach to this addresses the first problem, but not the second. A Depth-Limited Search (DLS)[11] is used to find a path that collects all tokens. Because a Depth First Search can easily go into an infinite recursion, using a DLS instead fixes that problem, while maintaining essentially the same behavior. The advantages of using this type of search is that the search itself is very fast and as such can be run consistently by the agent without impairing its performance.

Since an attempt to solve the second problem might affect the consistency with which the agent can make a new search, it would be preferable to process the best way to get from one platform to any other at the start of the level and then save that information to be used during the rest of the execution. This is accomplished by using a variation of Dijkstra's algorithm[12] to find the shortest path between every two platforms. Because ease of traversal is more important than pure optimization of length traveled, the weights of the different edges don't correspond to the real distances. Instead, to make bigger gaps be considered more costly than traversing two small gaps, the weights used are:

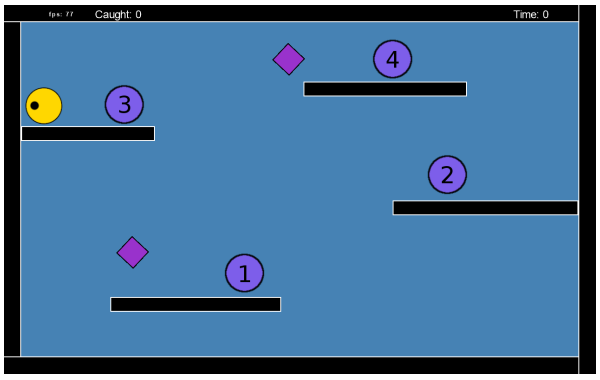


Figure 4: In this level, the agent can jump from the ground directly to platform 2. But that jump is at the very limit of the circle's capabilities, and it is much easier and reliable to jump to platform 1 first and then make the jump from there.

$$\max(\Delta x \log(\Delta x + 1), \Delta y \log(\Delta y + 1)) \qquad (1)$$

Where $\Delta x$ and $\Delta y$ are the horizontal and vertical lengths that separate the two platforms. This is done mostly to address the problem depicted in Figure 4.

With the best way to go from one platform to another being now known at all times, the plan now only needs to be the set of tokens that must be caught and in the correct order. The search was changed to reflect this, and to optimize the results yielded it is now a Uniform Cost Search(UCS)[13] rather than a DLS.

### D. Exploration versus Exploitation

Far from being as important as the other aspects in this section, the exploration/exploitation ratio is not so intimately related with the performance of the agent. It is, however, related to the consistency of that performance, as when the agent explores, its performance should be lower than when the agent is exploiting its knowledge. Since in this setup the levels in which the agent is learning and the levels in which the agent is tested are never the same (at the same time), it does not make much sense to allow the agents to explore in the levels when it is being tested. This is because the lessons the agent learns are only taken into account at the end of the level, so any exploration in a given level will not contribute to its completion at that time.

All of these components were then adapted to the rectangle character, the greatest difference being the need of the features to track the shape of the rectangle in each situation, by storing the rectangle's height at any given time.

## V. Emergence of Cooperation

Emergent behavior is usually relatively complex behavior that emerges from the simpler interactions between agents of a system. It is sometimes the aim of multi-agent systems to create the conditions for this type of behavior to emerge, or simply to see what kind of behavior emerges from relatively simple behaviors of individual agents. One example that illustrates the concept of emergence very well is Conway's Game of Life, where a set of very simple rules can lead to extremely complex results[14].

The need for emergent behavior comes from the need to have agents for the game that can play as well together as they play with other players, be they human or AI as well. Having the agents internally sharing information means that they are relying on a crutch that will not be there if they are playing alongside a different player. Agents that should be able to successfully play along with many different players, be they humans or other artificial agents that for whatever reason are not familiar with the inner workings of the first agent. To achieve this goal, the cooperative behavior of the agent cannot be tied directly to the behavior of another.

The first thing to try is to see how the agents perform in the cooperative levels. While it is naive to think that given their current states the agents will be able to apprehend anything new in a cooperative environment, this exercise also serves as a baseline for further work in these levels.

The next step in creating a cooperative solution is to change the features to include information about the other character. This small change, however, carries a heavy cost. Since the feature vector works as the identifier of a state, and those identifiers do not allow for partial matching of states, any training done with a different set of features immediately becomes obsolete when the features are changed.

To avoid having to completely get rid of past learning, the features only track the other character when it is relatively close. That way, on single player levels and when the two characters are far apart, it is still possible to make use of past learning. This does, however, make it harder for the agents to learn to operate in cooperative settings, as it reduces the amount of cooperative situations encountered even if all the training takes place in cooperative settings.

Although the agents cannot be expect to do much more than staying out of each other's way when trying to collect the tokens, there are limits to what can be done in the current solution, as changes to the agents' behavior have to either change the features or change the planning. It does not really leave room to add a new layer of behavior that is aimed at a different type of problems.

Table 1: Baseline GF Competition Circle Track 2014

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 1.35 | (2) | 12.20 | (20) | 65 | 140 |
| 02 | 1.70 | (3) | 45.00 | (45) | 0 | 170 |
| 03 | 1.20 | (3) | 58.20 | (60) | 5 | 120 |
| 04 | 2.63 | (4) | 76.47 | (80) | 10.53 | 263 |
| 05 | 1.00 | (2) | 70.00 | (70) | 0 | 100 |
| 06 | 1.20 | (2) | 30.50 | (40) | 35 | 121 |
| 07 | 1.00 | (3) | 60.00 | (60) | 0 | 100 |
| 08 | 2.11 | (3) | 38.79 | (40) | 26.32 | 210 |
| 09 | 2.05 | (3) | 77.16 | (80) | 10.53 | 205 |
| 10 | 0.53 | (3) | 100.00 | (100) | 0 | 52 |

Total Score: 1481

Table 2: Baseline GF Competition Circle Track 2015

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 1.10 | (2) | 28.40 | (30) | 10 | 110 |
| 02 | 1.58 | (3) | 60.00 | (60) | 0 | 157 |
| 03 | 1.00 | (3) | 90.00 | (90) | 0 | 100 |
| 04 | 2.45 | (4) | 88.30 | (90) | 10 | 245 |
| 05 | 2.65 | (4) | 90.00 | (90) | 0 | 265 |

Total Score: 877

## VI. Results and Discussion

The method of testing used to evaluate this solution is based on the way the participants on the GF competition are evaluated. All the tests with the results presented in this section were run in the same machine for reliability. In each individual run of each level, the number of tokens collected is counted and the agents are timed. At the end, using both these values, the score for that level is calculated using the same values as the competition.

$$SCORE = V_{completed} \times \frac{(T_{max} t)}{T_{max}} + (V_{collect} \times N_{collect}) \quad (2)$$

$V_{collect}$ is the score attributed to the agent (or team) for each collected diamond, $N_{collect}$ is the total number of diamonds collected within the time limit, $V_{completed}$ is a bonus score attributed to the agent (or team) if it successfully completes the level (e.g. collects all diamonds) within the time limit, and $t$ is the time it took the agent (or team) to complete the level. $T_{max}$ is the time limit of the level. Using the scores to compare performance between agents is the simpler and easier way to understand, even if looking at the actual values for tokens caught and time taken might provide better insight. Quitrio's agent was used as a baseline to test the agents again. While the baseline only provides values for the circle agent and that is the most reliable comparison, it is still possible to determine whether or not the rectangle agent and the cooperative performance can match the circle's.

Since there were multiple stages in improving the base solution, at each stage the agent has to be trained and tested to see how each change affected its performance. These changes are cumulative, and every time something is added or changed, the changes that came before are maintained and the whole set is tested. The first stage were the changes to the training set. There was a slight improvement on the 2015 levels, but overall there seems to be a deterioration, since in the 2014 set it performed worse. Still, the difference between the scores doesn't seem to be enough to draw any conclusions on whether or not there was a deterioration, especially because the performance of the agent is not always consistent.

Next come the new set of features. Considering how much they affect the learning process of the agent, coupled with the extensive changes that were made, the changes to the feature vector should be expected to have the most impact on the performance of the agent overall. Yet looking the results once again don't show significant changes in performance. Compared to the baseline, the performance on the 2014 levels was again slightly worse and the performance on the 2015 levels slightly better, while the total score was exactly the same. The simplest explanation for the similarity in results is that the changes made to the agent are not improvements and the new components are equivalent to the ones that they replaced. However looking at the extent of the changes, this does not seem very likely.

The next stage is testing the changes made to the planning process. While ideally the changes to the planning would be tested alone, there were two additional minor changes that happened in the gap between testing the new features and testing the planning. The first one is the aforementioned change to the exploration and exploitation ratio and the second is a small change to the agent's reactive component. Considering how the planning interacts with the other components of the agent, the expectation is that the changes to the planning should only make the agent more efficient, since they don't improve the agent's ability to navigate the levels. It is true that the focus on ease rather than just minimizing the distance traveled can affect the paths the agent chooses, and if the agent is choosing easier paths, it just might start succeeding in places where it once failed.

For the first time there is a significant improvement in the agent's performance. Considering the changes this time were not limited to changes in the planning, it is very well possible that the other changes had some impact as well. This stage also included the extrapolation of all the components for the rectangle agent, which performed well below the baseline for the circle.

The final stage of the evaluation is assessing how well the agents perform in the cooperative levels. To solve cooperative levels, two different approaches

Table 3: No Training in GF Competition Circle Track 2014

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 2.00 | (2) | 7.70 | (20) | 100 | 206 |
| 02 | 1.90 | (3) | 45.00 | (45) | 0 | 190 |
| 03 | 2.00 | (3) | 46.50 | (60) | 30 | 200 |
| 04 | 2.70 | (4) | 72.90 | (80) | 30 | 270 |
| 05 | 1.00 | (2) | 70.00 | (70) | 0 | 100 |
| 06 | 1.00 | (2) | 40.00 | (40) | 0 | 100 |
| 07 | 0.80 | (3) | 60.00 | (60) | 0 | 80 |
| 08 | 2.60 | (3) | 31.50 | (40) | 80 | 261 |
| 09 | 2.90 | (3) | 51.00 | (80) | 90 | 293 |
| 10 | 1.50 | (3) | 93.60 | (100) | 10 | 150 |

Total Score: 1850

Table 4: No Training in GF Competition Circle Track 2015

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 1.00 | (2) | 30.00 | (30) | 0 | 100 |
| 02 | 2.11 | (3) | 58.22 | (60) | 11.11 | 211 |
| 03 | 1.00 | (3) | 90.00 | (90) | 0 | 100 |
| 04 | 3.50 | (4) | 70.10 | (90) | 70 | 351 |
| 05 | 3.60 | (4) | 78.70 | (90) | 60 | 360 |
| 06 | 2.90 | (3) | 27.20 | (40) | 90 | 292 |
| 07 | 1.00 | (2) | 60.00 | (60) | 0 | 100 |
| 08 | 2.10 | (4) | 60.00 | (60) | 0 | 209 |
| 09 | 0.00 | (3) | 40.00 | (40) | 0 | 0 |
| 10 | 0.00 | (2) | 50.00 | (50) | 0 | 0 |

Total Score: 1723

were tried, one using the same methods the agents already used and the other by adding new fields to the feature vector. Both approaches are consequence of the limitations imposed by the current internal representation and both approaches should not be expected to do more than allow the agents to avoid getting in each other's way, if they even manage that.

The first approach is the more naive one, where the agents used the previous training in individual levels and extended it with cooperative levels. Since in these circumstances the agents are not keeping track of the other one, it is very likely that the agents will perform poorly in the cooperative levels. Both approaches have similar results, which seems to suggest that the features aren't being as relevant to the performance as would be expected. In fact, testing the agents without any learning at all also yields very similar results. The results of the final version are as follows:

Ultimately there was a significant improvement in the circle's performance, while the other aspects fared much worse. This seems to suggest that the reactive component has the most responsibility over the results of the agents.

## VII. Conclusions and Future Work

The goal of this work was to extend already existing agents for the Geometry Friends game in order to give them the ability to work together. Because the aim was also to create agents that can cooper-

Table 5: No Training in GF Competition Rectangle Track 2014

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 0.80 | (2) | 40.00 | (40) | 0 | 80 |
| 02 | 0.10 | (2) | 25.00 | (25) | 0 | 10 |
| 03 | 0.80 | (3) | 80.00 | (80) | 0 | 80 |
| 04 | 0.20 | (2) | 20.00 | (20) | 0 | 20 |
| 05 | 0.20 | (5) | 90.00 | (90) | 0 | 20 |
| 06 | 1.33 | (3) | 40.00 | (40) | 0 | 133 |
| 07 | 1.80 | (3) | 50.00 | (50) | 0 | 180 |
| 08 | 0.00 | (3) | 60.00 | (60) | 0 | 0 |
| 09 | 1.33 | (3) | 35.00 | (35) | 0 | 133 |
| 10 | 0.00 | (3) | 35.00 | (35) | 0 | 0 |

Total Score: 656

Table 6: No Training in GF Competition Rectangle Track 2015

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 1.44 | (3) | 30.00 | (30) | 0 | 144 |
| 02 | 0.00 | (3) | 60.00 | (60) | 0 | 0 |
| 03 | 1.11 | (3) | 60.00 | (60) | 0 | 111 |
| 04 | 1.50 | (4) | 90.00 | (90) | 0 | 150 |
| 05 | 0.30 | (4) | 90.00 | (90) | 0 | 30 |
| 06 | 0.67 | (2) | 20.00 | (20) | 0 | 66 |
| 07 | 0.00 | (3) | 60.00 | (60) | 0 | 0 |
| 08 | 0.00 | (4) | 50.00 | (50) | 0 | 0 |
| 09 | 1.70 | (2) | 23.90 | (30) | 70 | 171 |
| 10 | 0.40 | (2) | 44.50 | (45) | 10 | 40 |

Total Score: 712

Table 7: No Training in GF Competition Cooperative Track 2014

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 0.00 | (3) | 90.00 | (90) | 0 | 0 |
| 02 | 0.70 | (3) | 90.00 | (90) | 0 | 70 |
| 03 | 1.00 | (2) | 35.00 | (35) | 0 | 100 |
| 04 | 0.30 | (5) | 110.00 | (110) | 0 | 30 |
| 05 | 1.00 | (4) | 100.00 | (100) | 0 | 100 |
| 06 | 3.00 | (4) | 90.00 | (90) | 0 | 300 |
| 07 | 0.78 | (2) | 60.00 | (60) | 0 | 77 |
| 08 | 0.00 | (2) | 90.00 | (90) | 0 | 0 |
| 09 | 0.20 | (3) | 55.00 | (55) | 0 | 20 |
| 10 | 0.00 | (2) | 35.00 | (35) | 0 | 0 |

Total Score: 697

Table 8: No Training in GF Competition Cooperative Track 2015

| Level | Tokens | (Max) | Time | (Max) | Sucess(%) | Score |
|---|---|---|---|---|---|---|
| 01 | 0.10 | (4) | 90.00 | (90) | 0 | 10 |
| 02 | 1.00 | (2) | 30.00 | (30) | 0 | 100 |
| 03 | 0.10 | (3) | 60.00 | (60) | 0 | 10 |
| 04 | 1.00 | (3) | 60.00 | (60) | 0 | 100 |
| 05 | 0.00 | (3) | 90.00 | (90) | 0 | 0 |
| 06 | 2.00 | (3) | 40.00 | (40) | 0 | 200 |
| 07 | 1.10 | (5) | 90.00 | (90) | 0 | 110 |
| 08 | 0.00 | (2) | 90.00 | (90) | 0 | 0 |
| 09 | 0.00 | (2) | 25.00 | (25) | 0 | 0 |
| 10 | 0.00 | (2) | 40.00 | (40) | 0 | 0 |

Total Score: 530

ate with human players or even other agents, this ability could not be tied directly to the other agent and has to emerge from the agent's other abilities.

The individual results of the circle agent are quite satisfactory since its results are on par with other agents submitted for the competition in the same tracks. The results of the rectangle, however, are not so good. While it is possible that when extrapolating the circle agent's functionality it wasn't properly adapted to the unique problems the rectangle agent faces, it seems more likely that the lack of a well developed reactive component that allowed for adequate exploration is actually what caused the underwhelming results.

In fact, the reactive component seems to be of much greater consequence than anticipated, either because it is crucial for the agent's ability to explore during the training, or even because the learning process isn't working quite as intended. It is possible that the reactive component just has too great a weight in the results yielded by the learning process but it is also possible that the learning is failing to match the current state with one from the learning more often than expected and the agent is defaulting to its reactive behavior as a consequence.

Comparing the performance of the trained agents against the untrained agents, the second option seems more likely to be true. Either way, the fact is that the learning is not working as intended or expected, and that limits what is possible to achieve with this method.

In order to make the most out of the ideas used so far, changing the internal representation of the learning is unavoidable. As it is, it seems likely that the learning will only work in situations that were previously encountered. Even the attempts to generalize these situations probably only resulted in reducing the resolution of the agent's representation of the world, rather than actually allowing the agents to generalize or extrapolate to new situations.

One possibility that addresses a lot of issues is a decision tree. The knowledge gathered so far can be used to build the decision tree. This knowledge can then be extended and corrected as the agent continues to go through the levels. The very process of generating the decision tree may be enough to select the relevance of each of the features, and if the absence of a certain feature is simply considered as another possible value for it, it then becomes possible to support different configurations of features for the same decision tree. A new feature would not be of any worth when it is introduced, but as it comes into play more and more often it will eventually integrate the tree as well. As such, more and more knowledge can be added to the world representation without reducing the capacity to generalize.

Even with these changes, creating a competent reactive component for the rectangle agent will probably still be necessary. With a more robust learning system the bias created by the reactive component should not be so relevant, and if the agent has a naive approach to solving the levels rather than a mostly random one, it should considerably speed up the learning process.

Another problem of the rectangle agent is how it is handling its different shapes. It is likely that the navigational graph for the rectangle will need to take into account what shape the rectangle should be in for each connection between platforms.

Finally, the separation into the three sub problems may need to be revisited, as it doesn't seem to agree very much with cooperative play. In the case of the circle, things can be broken down to the circumstances of the jump, what position is ideal, what speed should the character have at that time of the jump, maybe even plan ahead to determine the optimal circumstances for the jump and then creating those circumstances in preparation and the rectangle agent should prove to be even simpler.

With these changes the agents should be then in a position where they can play with at least some success with both other agents and human players.

**References**


[1] R. Prada, P. Lopes, J. Catarino, J. Quitrio, F. Melo, "The Geometry Friends Game AI Competition", textitproceedings of CIG2015 IEEE Conference on Computational Intelligence and Games, pp. 431-438 (2015)

[2] P. Stone, G. Kaminka, S. Kraus, J. Rosenschein, "Ad Hoc Autonomous Agent Teams: Collaboration without Pre-Coordination", *AAAI* (2010)

[3] E. Pagello, A. D'Angelo, F. Montesello, F. Garelli, C. Ferrari, "Cooperative behaviors in multi-robot systems through implicit communication" *Robotics and Autonomous Systems*, pp. 65-77 (1999)

[4] M. Bowling, P. McCracken, "Coordination and Adaptation in Impromptu Teams", *AAAI*, pp.53-58 (2005)

[5] A. M. Turing, "Computing machinery and intelligence", *Mind*, pp. 433460 (1950)

[6] F. Hsu "Behind Deep Blue: Building the Computer that Defeated the World Chess Champion", *Princeton University Press* (2002)

[7] J. Togelius, S. Karakovski, R. Baumgarten, "The 2009 Mario AI Competition", *Proceedings of the IEEE Congress on Evolutionary Computation* (2010)

[8] P. Hingston, "A Turing Test for Computer Game Bots", *Computational Intelligence and AI in Games, IEEE Transactions on*, pp. 169-186 (2009)

[9] C. Fraga, "Motion Control for an Artificial Character teaming with a Human Player in a Casual Game", *Master Thesis* (2011)

[10] J. Quitrio, "A Reinforcement learning approach for the circle agent of Geometry Friends", *Master Thesis* (2015)

[11] S. Russel, P. Norvig, "Artificial Intelligence: A Modern Approach (2nd Edition)", *Prentice Hall*, p.78 (1995)

[12] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs",*Numerische Mathematlk*, pp. 269-271 (1959)

[13] S. Russel, P. Norvig, "Artificial Intelligence: A Modern Approach (2nd Edition)", *Prentice Hall*, pp.76-77 (1995)

[14] M. Gardner, "Mathematical games: The fantastic combinations of John Conways new solitaire game 'life'", *Scientific American vol. 223*, pp. 120-123 (1970)