2013 Team Description Paper: UBC Thunderbots Simultation

Terence Yen-Yi Lin[a], Devon Ash

Departments of: (a) Computer Science,
The University of British Columbia
2329 West Mall, Vancouver, BC Canada V6T 1Z4
ubcthunderbots.ca
robocup@ece.ubc.ca

Abstract. This paper details the 2013 design of UBC's 2D Simulation League team, to be entered at RoboCup 2013 in Eindhoven, The Netherlands. The main focus of this year for the team is to restart UBC's efforts in the Robocup 2D Simulation League.

1 Introduction

UBC Thunderbots Simulation is a team of Computer Science graduate and undergraduate students at the University of British Columbia. Established in late 2012, it is pursuing its first competitive initiative within the 2D Simulation League at RoboCup 2013.

The team has its roots in UBC Thunderbots, which was formed in 2006 and have been competing in the Robocup Small Size League (SSL) since 2009. UBC Thunderbots decided to venture into the 2D simulation league after some research were made on the Keepaway Problem [3] to improve the team's performance. As such, Thunderbots Simulation uses approaches and techniques found in SSL and the Keepaway problem to improve over the code base being used. Research in competitive robotic soccer in UBC can be further traced back to Prof. Alan Mackworth, who still provides general supervision to the team.

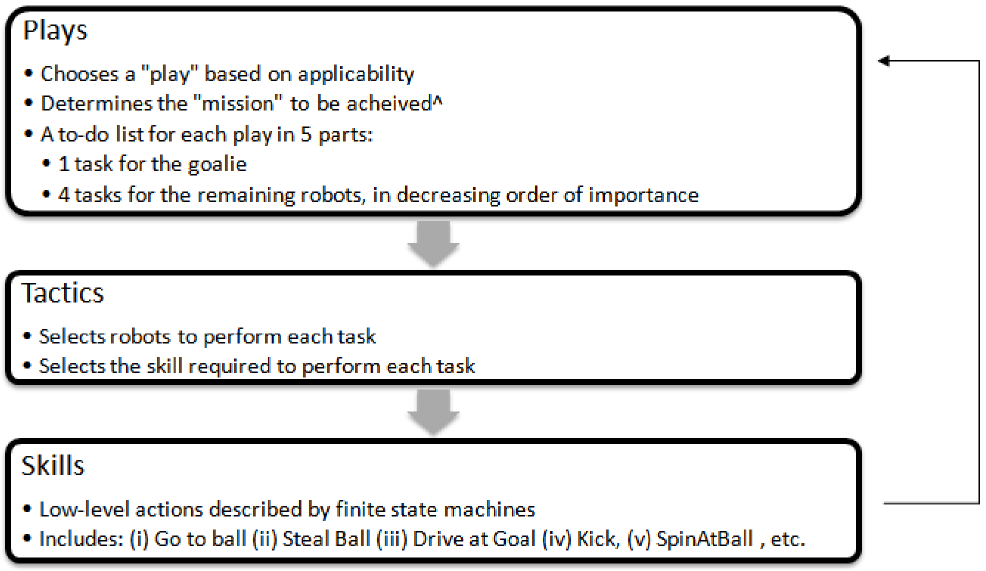
The remainder of this paper is organised as follows. Section 2 describes the Skills, Tactics, and Plays (STP) architecture approach developed by CMDragons [2]. Section 3 presents the Tile Coding techniques used by our Reinforcement Learning Algorithms. Section 4 briefly describes the Reinforcement Learning Algorithm being used with some empirical results. Finally, the paper is concluded in Section 5.

The team's source code is based on Agent2D base, version 3.1.1, developed by H. Akiyama [4] of Team Helios.

2 Skills, Tactics, and Plays

The main high-level decision making model used is the STP model, developed by CMDragons in 2003 [2] for the Robocup Small Size League competitions.

The STP model is used to manage multiple robots in a challenging adversarial environment. In this environment, the decision making model handle short dynamic events, while simultaneously trying to achieve long-term objectives. STP is composed of Skills, for executions of low-level actions, such as a simple move or kick; Tactics, that determine the skills for use by the robots; and Plays, which assign roles for the robots to execute tactics [1]. The hierarchical architecture within the STP model allows for dynamic quick response and coordinated control. The team will be able to achieve long-term goals in a highly coordinated manner, and, similarly, reacts to dynamic events initiated by the enemy.



Plays
• Chooses a "play" based on applicability
• Determines the "mission" to be acheived^
• A to-do list for each play in 5 parts:
• 1 task for the goalie
• 4 tasks for the remaining robots, in decreasing order of importance
Tactics
• Selects robots to perform each task
• Selects the skill required to perform each task
Skills
• Low-level actions described by finite state machines
• Includes: (i) Go to ball (ii) Steal Ball (iii) Drive at Goal (iv) Kick, (v) SpinAtBall , etc.

Fig. 1: STP Model. Note: SSL had 5 players for each team, but STP is easily extendable to have any number of players.

Going back to the Agent2D code base, the three layers of STP, Skills, Tactics, and Plays, can roughly be translated respectively into Behaviors, Roles, and Formation layers of abstraction. Currently the main focus of improvement over Agent2D is in the individual Roles to enable then to better select Behaviors to perform through the use of Reinforcement Learning.

In the future, the team also wishes to improve the current base Agent2D strategy with the mentioned Play or Formation selection model in the diagram above.

3 Tile Coding

Before jumping into the details of how Reinforcement Learning is being used, one should look at the technique being used to achieve efficient Reinforcement Learning.

The success of Reinforcement Learning on real-world problems with large, often continuous state and action spaces, especially a problem like soccer where multiple agents need to be play both cooperatively and competitively, is highly dependent on effective and efficient function approximation.

In practical applications of Reinforcement Learning, States and Actions are defined by continuous variables such as distances and angles. As a result, the Q-table for tuple of States and Actions are typically large or infinite, thus learning the value function requires some form of function approximation to reduce the size of the Q-table. [5].

Tile Coding does this function approximation for us by representing the values of a vector of continuous variables as a large binary vector with few 1s and many 0s. The binary vector is not represented explicitly, but as a list of the 1s in the vector. The main step is to partition, or tile, the continuous space multiple times and select one tile from each tiling corresponding to the vector's value. Each tile is converted (hashed to) an element in the large binary vector, and the list of the tile numbers is returned as the representation of the vector's value [6].

An immediate advantage of Tile Coding is that the overall number of features that are present at any time is strictly controlled and independent of the input state. Exactly one feature is present in each tiling, thus the total number of features present is always the same as the number of tilings. Tile coding thus strikes a successful balance among representational power, computational cost, and ease of use [5] [6].

4 Reinforcement Learning with Tile Coding

The Reinforcement Learning algorithm chosen to use Tile Coding for function approximation is SARSA, with or without Eligibility Trace [7]. An eligibility trace is a temporary record of the occurrence of an event, such as the visiting of a state or the taking of an action. The trace marks the memory parameters associated with the event as eligible for undergoing learning changes. If an error occurs, only the eligible states or actions are assigned credit or blame for the error (receive positive or negative rewards). Thus, eligibility traces help bridge the gap between events and training information [7].

Each Role uses SARSA to learn to pick the best Behavior (provided by the Agent2D base) to use given the state. Although each Role can be configured to use different variables for the state representation necessary to run SARSA, the general state is represented with the variables as followed:

-Ball kickable of not (1 or 0);
-There is a teammate who can kick the ball (1 or 0).
-Distance of closest teammate to the ball to the ball
-Distance of player to the ball
-Distance to player to the closest teammate to the ball
-Distance of the ball to our goal
-Distance of the player to our goal
-Angle of the player to the ball

-Angle of the closest player to the ball to the ball

The reward function is a simple combination of scoring and distance of the ball to our goal. The reward is highly weighted towards scoring. If our team scores the agent is rewarded postively, and if the enemy scores negative reward is received.

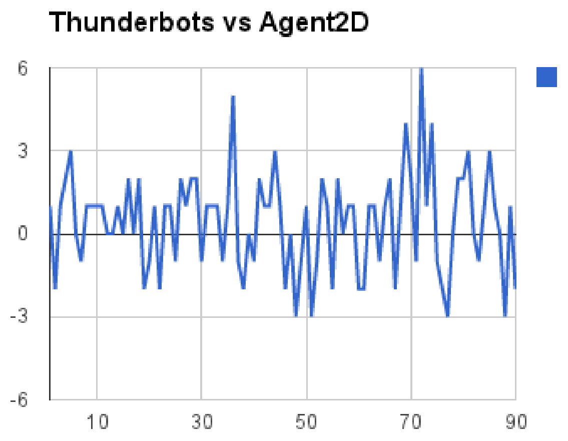


Fig. 2: Thunderbots vs Agent2D, (Thunderbots score - Agent2D score) per game.

With its current configuration, on average Thunderbots was able to achieve 0.44 more score per game, giving it an edge over its predecesor Agent2D thanks to the Reinforcement Learning it was given.

5 Conclusion

This paper introduced our RoboCup soccer 2D simulation team, UBC Thunderbots Simulation, and described our current research efforts, including: 1) Introduction of the STP model from SSL to Simulation League, 2) Tile Coding and its use in 3) Reinforcement Learning algorithms such as SARSA with Eligibility trace. The empirical results also shows that Thunderbots has improved over its Agent2D origin and could be competitive in the League.

6 Acknowledgements

We'd like to thank the Faculty of Science and Applied Science; the departments of Computer Science and Mechanical Engineering; and the University of British Columbia for their continued support. We'd also like to thank all our sponsors for their generous contributions to graduate and undergraduate student learning and research. Special thanks to Prof. Alan Mackworth for his supervision and support for the team.

References

1. Browning, B., Bruce, J., Bowling, M., Veloso, M. *STP: Skills, tactics and plays for multi-robot control in adversarial environments*, 2004.
2. Browning, B., Bruce, J., Bowling, M., Veloso, M. *CMDragons03 Team Description Paper*, 2003.
3. Stone, P. *Learning to Play Keepaway*, online, available athttp://www.cs.utexas.edu/ AustinVilla/sim/keepaway/, consulted 2012.
4. Akiyama, H. *Agent2D*, online, available at http://rctools.sourceforge.jp/, consulted 2012.
5. Sheratov, A., Stone, P. *Function Approximation via Tile Coding: Automating Parameter Choice*, 2005.
6. Sutton, R. *Tile Coding Software*, online, available at http://webdocs.cs.ualberta.ca/ sutton/tiles2.html, consulted 2012.
7. Sutton, R. *Eligibility Traces*, online, available at http://webdocs.cs.ualberta.ca/ sutton/book/ebook/node72.html, consulted 2013.