In proceedings of the AAMAS 2016 workshop on Adaptive Learning Agents
May 2016

Half Field Offense: An Environment for Multiagent Learning and Ad Hoc Teamwork

Matthew Hausknecht
University of Texas at Austin
mhauskn@cs.utexas.edu

Prannoy Mupparaju
IIT Bombay
m.prannoy@iitb.ac.in

Sandeep Subramanian
IIT Bombay
110260028@iitb.ac.in

Shivaram Kalyanakrishnan
IIT Bombay
shivaram@cse.iitb.ac.in

Peter Stone
University of Texas at Austin
pstone@cs.utexas.edu

ABSTRACT

The RoboCup 2D simulation domain has served as a platform for research in AI, machine learning, and multiagent systems for more than two decades. However, for the researcher looking to quickly prototype and evaluate different algorithms, the full RoboCup task presents a cumbersome prospect, as it can take several weeks to set up the desired testing environment. The complexity owes in part to the coordination of several agents, each with a multi-layered control hierarchy, and which must balance offensive and defensive goals. This paper introduces a new open source benchmark, based on the Half Field Offense (HFO) subtask of soccer, as an easy-to-use platform for experimentation. While retaining the inherent challenges of soccer, the HFO environment constrains the agent's attention to decision-making, providing standardized interfaces for interacting with the environment and with other agents, and standardized tools for evaluating performance. The resulting testbed makes it convenient to test algorithms for single and multiagent learning, ad hoc teamwork, and imitation learning. Along with a detailed description of the HFO environment, we present benchmark results for reinforcement learning agents on a diverse set of HFO tasks. We also highlight several other challenges that the HFO environment opens up for future research.

Categories and Subject Descriptors

H.4 [**Computing methodologies**]: Multi-agent systems

General Terms

Algorithms, Measurement, Design

Keywords

Half Field Offense, RoboCup, Ad-Hoc Teamwork, Reinforcement Learning

1. INTRODUCTION

For agents to act with greater autonomy, it is crucial that they learn from their experience, which is often shared with other agents. These other agents could themselves be cooperative partners, adversaries, or teachers. It is no surprise, then, that the quest to design autonomous agents has spawned several fields of study to investigate these essential aspects of agent behavior. The field of **reinforcement learning** [22] examines how agents in an unknown environment, through trial and error, can learn to take actions with long-term benefit. **Imitation learning** [3] specifically considers how the learning process can be sped up by harnessing instructive advice from a teacher. Whereas the predominant body of work under these topics focuses on the single-agent setting, **multiagent reinforcement learning** [15, 24] has taken shape as an active area of research in its own right. Other topics of interest in a multiagent environment, such as **coordination** [23] and **ad hoc teamwork** [20] have also been actively pursued.

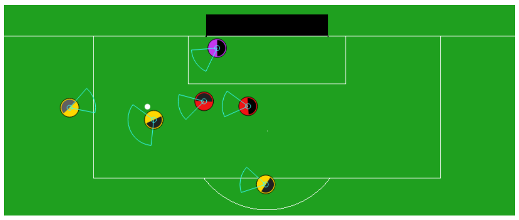

Figure 1: 3v3 Half Field Offense: Yellow offense agents search for an opening in the defensive formation. Red defenders and purple keeper strive to intercept the ball or force it out of bounds. HFO is better understood by video than picture: 1v1 `https://vid.me/sNev`, **2v2** `https://vid.me/JQTw`, **3v3** `https://vid.me/1b5D`

In each of the areas listed above, significant progress has been made in establishing theoretical foundations and conceptual frameworks. However, the validation of the resulting algorithms has typically been in constrained settings that do not possess the full complexity of the real world. For example, reinforcement learning algorithms are most often tested on toy problems such as Mountain Car and Acrobot, if not in small, discrete, "grid world" environments [22], which have also been used in several multiagent studies [11, 24]. Naturally there are merits to testing an algorithm in a simplified environment that does not include orthogonal or confounding factors. On the other hand, such factors are bound to present themselves when the algorithm is taken to the real world. In fact, real-world applications may additionally demand the integration of ideas from different fields of study, thereby motivating the need for test environments that afford such a possibility.

This paper accompanies the release of an environment for benchmarking algorithms related to learning, multi-agency, and teamwork. Our environment is built on top of the RoboCup 2D simulation platform [1]. RoboCup [13] is an international robot soccer competition that promotes research in AI and robotics. Within RoboCup, the *2D simulation league* works with an abstraction of soccer wherein the players, the ball, and the field are all 2-dimensional objects. For nearly two decades now, 2D simulation soccer has fostered active research and development. However, for the researcher looking to quickly prototype and evaluate different algorithms, the full soccer task presents a cumbersome prospect: full games are lengthy, have high variance in their outcome, and demand specialized handling of rules such as free kicks and offsides.

Our objective is to expose the experimenter only to core decision-making logic, and to focus on the most challenging part of a RoboCup 2D game: scoring and defending goals. To this end, we introduce the Half Field Offense (HFO) environment (Figure 1). As a machine learning task, HFO features a diversity of challenges: in the simplest form, HFO requires the development of a single controller for an autonomous 2D soccer agent. This agent could be either playing offense and seeking to score goals or playing defense and acting to prevent goals. Beyond single-agents, HFO supports multiple agents, some of which may be manually controlled by the user and others that can be automatically controlled. Thus HFO incorporates aspects of multiagent learning and ad hoc teamwork [20]. HFO naturally lends itself to reinforcement learning due to the sequential nature of the decisions made by the agents. The environment involves a continuous state space, and provides a choice between continuous and discrete action spaces.

Indeed HFO was originally introduced by Kalyanakrishnan *et al.* [12] almost a decade ago, but the authors did not release code for their framework. Barrett and Stone [6] recently reported some experiments on an independently-developed code base for HFO, which again was not released publicly. Thus HFO has remained inaccessible to many potential users.

Among publicly-released benchmarks for multiagent RL, the closest in spirit to HFO is Keepaway [21], which models the task of ball possession in soccer. Note that possession is only *one* of the many skills an HFO team must master, in addition to moving towards the goal and shooting. Also, while intermediate rewards are natural to define in Keepaway, credit is only available at the end of an episode in HFO. These reasons make Keepaway an easier task for learning than HFO [12]. Our public release of HFO also shares the same motivations as the recently-released Arcade Learning Environment [7], which provides easy access to a large number of console games. However, these games are all in the single-agent setting.

Our open-source release of the HFO environment brings several convenient features, as listed below.[1]

- Standard, MDP-like interface to RoboCup server.
- Access to high and low-level state spaces.
- Access to high and low-level action spaces.
- Support for automated teammates and opponents.
- Ability to play offense or defense.
- Facilities for inter-agent communication.
- Setup to perform reproducible experiments.
- Tools for performance evaluation and analysis.

[1]Repository hosted at `https://github.com/LARG/HFO`.

The remainder of this paper is organized as follows. Section 2 presents the Half Field Offense environment in detail. Section 3 provides an overview of different agents that we have benchmarked in this environment, and Section 4 reports the corresponding benchmark results. In Section 5, we outline some of the challenges that the HFO environment opens up to future research. Section 6 discusses related work, and Section 7 serves as the conclusion.

2. HALF FIELD OFFENSE

Competition RoboCup 2D soccer is played between two teams of autonomous agents who communicate with a central soccer server. The HFO Environment (Figure 2) builds upon the competition-ready RoboCup 2D server, but supports smaller teams consisting of arbitrary mixes of automated teammates (NPCs, for "non player characters") and player-controlled agents - up to ten players per side.

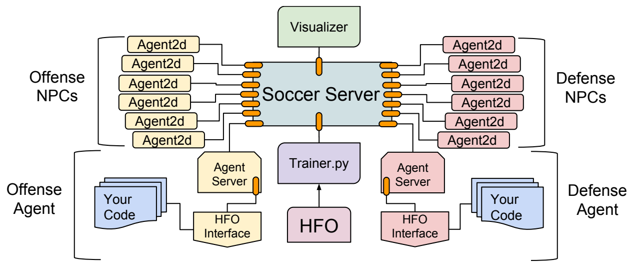

Figure 2: The HFO Environment is comprised of many separate processes which communicate over the network with the RoboCup 2D soccer server. HFO starts these processes, ensures they communicate, and oversees the games. A user needs only to specify the number of offensive and defensive agents and NPCs and then connect their agent(s) to the waiting Agent Server(s) via the HFO interface.

Because the official RoboCup 2D soccer server lies at the core of the HFO Environment, we expect agents or skills learned in HFO will translate with relatively little effort into competition RoboCup soccer. Additionally, cutting-edge RoboCup competition-winning agents can be ported into HFO as NPCs. Currently, HFO supports one type of teammate—Helios-Agent 2D—but standard communication interfaces allow others to be integrated easily. The next section discusses the state and action spaces provided by the HFO Environment.

2.1 State Representation

Agents interfacing with the HFO domain choose between a low or high level state representation. This choice of representation affects the difficulty of the HFO task. The low-level representation provides more features with less preprocessing, while the high-level representation provides fewer, more informative features. For example, the high-level representation provides a feature for the largest open goal angle (Figure 3), computed by comparing the keeper's position to the positions of the goal posts. In contrast, the low-level representation provides angles to the goal posts and angles to each of the opponents, but determining which opponent is the keeper and calculating open goal angles is left to the player. We now describe these representations in more detail.

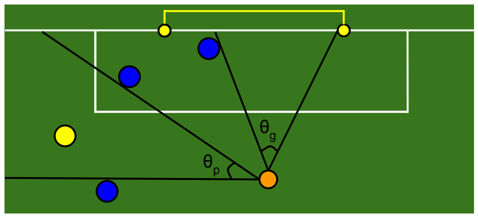

Figure 3: High-level state representation computes the orange agent's largest open goal angle θ_g as well as open pass angle to a teammate θ_p.

2.2 Low-Level State Representation

HFO defines a low-level egocentric viewpoint encoded using 58 continuously-valued features (8 additional features are added for each teammate and opponents). These features are derived through Helios' [2] world model and provide angles and distances to various on-field objects of importance such as the ball, the goal, and the other players. The most relevant features include the agent's position, velocity, orientation, and stamina; indicator if the agent can kick; angles and distances to the ball, goal, corners of the field; teammates and opponents positions, angles, velocities and orientations. These features are intended to be used by a learning algorithm that can handle raw perceptions.

2.3 High-Level State Representation

The high-level feature space is a compact representation enabling an agent to learn quickly by generalizing its knowledge about similar states. There are a minimum of 9 continuous features with an additional 5 for each teammate. Features include: agent's position and orientation; distance and angle to the ball; indicator if the agent can kick; distance and angle to the goal; largest open goal angle (θ_g in Figure 3); teammates distance, angle, uniform number, largest open goal angle and nearest opponent distance; open pass angle to each teammate (θ_p in Figure 3). This feature set is inspired by Barrett et al.'s exploration of HFO [5]. These high-level features allow a learning agent to quickly improve HFO performance.

2.4 Action Representation

The action space of a domain is one component of the complexity of the learning problem. HFO provides a choice between two action spaces - one high-level and discrete, the other low-level and continuous.

2.5 Low-Level Action Representation

HFO uses a parameterized low-level action space in which an agent is first faced with a decision about which type of action to select, and then how to execute that action. Formally characterized as a Parameterized Action Markov Decision Process (PAMDP) [17], HFO's low-level, parameterized action space features four mutually-exclusive action primitives: Dash, Turn, Tackle, and Kick. Each action has up to two continuous parameters which must also be specified. An agent must select both the discrete action it wishes to execute as well as the accompanying continuous parameters:

Dash(power, direction): Moves the agent in the indicated direction with requested speed. Movement is faster forward than sideways or backwards.
Turn(direction): Turns the agent in the indicated direction.
Tackle(direction): Slide tackles in the indicated direction.
Kick(power, direction): Kicks the ball in the indicated direction with requested power.

This low-level action space presents a challenge to learning algorithms. An agent acting randomly in the low-level state space will wander near its starting position, and is highly unlikely to approach the ball or score a goal.

2.6 High-Level Action Representation

HFO's high-level action space defines a compact interface to the Helios agent's soccer behaviors. Each high-level behavior is ultimately composed of low-level actions, but also incorporates Helios' strategy for choosing and parameterizing the low-level actions. HFO supports five high-level actions:

Move(): Moves the agent according to Helios' strategy.
Shoot(): Takes the best available shot.
Pass(uniform num): Passes to the teammate with the requested uniform number.
Dribble(): Advances the ball towards the goal using a combination of short kicks and moves.
Catch(): Goalie-specific action to capture the ball.

While **Shoot**, **Pass**, and **Dribble** are the choices available to an offense player, **Move** and **Catch** apply to defense players. Agents may also choose to do nothing by selecting a **NO-OP** action. Using this action space, an offensive agent that randomly select actions is capable of scoring goals as long as no keeper is present.

2.7 Automated Teammates (NPCs)

Automated teammates and opponents in HFO use a policy derived from Helios, the 2012 RoboCup 2D champion team [2]. This policy is designed for full 11-versus-11 matches, but gracefully scales to any of the smaller tasks in the HFO umbrella. As our benchmark results indicate, automated teammates and opponents using the Helios policy exhibit strong but not perfect policies. More importantly, Helios teammates favor cooperation and will strategically pass the ball to player-controlled agents. While some passes are direct, lead passes require the player-agent to quickly reposition in order to receive. When the player has the ball, Helios teammates intelligently position themselves and will sprint to receive a pass from the player.

2.8 Evaluation Metrics

Having presented the basic state spaces, action spaces, and NPCs featured in the HFO Environment, we now address the important question of how to evaluate the performance of HFO agents.

The HFO environment does not provide reward signals and instead indicates the ending status of the game. HFO episodes end with one of the following termination conditions:

Goal: The offense scored a goal.
Captured (CAP): The defense gained control of the ball.

Out of Bounds (OOB): The ball left the playfield.
Out of Time (OOT): No agent has approached the ball in the last 100 timesteps.

Using these termination conditions, we propose two evaluation metrics: Goal Percentage and Time to Goal. The primary focus of learning in HFO is to score goals when playing offense and prevent goals from being scored when playing defense. The primary metric, **Goal Percentage**, the percentage of all trials that end with a goal being scored, captures exactly this notion. The hallmark of an effective offensive agent is a high goal percentage. A second metric, **Time to Goal** (TTG), is defined as the number of timesteps required to score in each trial that culminates with a goal. Efficient offensive agents typically seek to minimize time to goal, while defenders strive to maximize this metric.

Finally, the HFO environment also indicates the last player to touch the ball. This information may be used to keep track of offensive passes and define alternative reward functions.

2.9 Learning Paradigms

The HFO Environment supports several learning paradigms: **Single-Agent Learning**, involves a lone offensive or defensive agent playing against one or many opponents. In **Ad Hoc Teamwork**, the agent must learn to cooperate with one or more unknown teammates without pre-coordinated strategies [5, 20]. In the case of HFO, learning agents have the opportunity to act as the ad hoc teammate of the Helios agents. Finally, **Multiagent Learning** places two or more learning agents on the same team with the shared objective of scoring or defending the goal. Known as Multiagent Reinforcement Learning (MARL), the challenge for these agents is to learn both individual competency as well as cooperation [24]. While not examined in this paper, HFO also supports configurations that blend these learning paradigms. For example, a team could consist of several learning agents paired with one or more Helios teammates, mixing multiagent learning with ad hoc teamwork. Additionally, HFO can create multiagent scenarios in which agents have competing objectives, for example by allocating some learning agents to play offense and others to play defense.

Having addressed the basic features of HFO, we now present benchmark agents designed for single-agent and multiagent learning, ad hoc teamwork.

3. BENCHMARK AGENTS

The HFO Environment makes it convenient to develop and deploy agents in different learning scenarios. Agent interfaces are provided for C++ and Python. Most benchmark agents are powered by reliable, well-understood learning algorithms that have withstood the test of time. We consider the following agents:

3.1 Random Agent

The **low-level random agent** randomly selects actions in the low-level action space and generates random continuous parameters to accompany these actions. Observed behavior is Brownian motion around the agent's starting position. This agent is excluded from the results as it never manages to score a goal or approach the ball.

In contrast the **high-level random agent** randomly selects actions in the discrete high-level action space. Observed behavior is erratic but eventually manages to score goals. Both agents serve as a lower bound for performance.

3.2 Hand-coded Agent

We designed a hand-coded agent that uses the high-level state action space described above. Its offensive policy, used both for scoring on an empty goal and for scoring on a keeper, first checks if the agent has the ball. If it does, and the distance to goal is less than α and the goal open angle is greater than β, the agent will Shoot; otherwise it will Dribble. If the agent does not have possession of the ball, it will take the Move action. This policy ensures the agent is close enough to the goal and has enough of an opening to put a shot through. Both α and β are optimized using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10].

A major difference between single agent learning and ad hoc teamwork is the availability of a teammate to pass and coordinate with. We modify the hand-coded policy to accommodate the Ad Hoc scenario by utilizing teammate-aware features: Now, when the agent has the ball, if it does not have an open shot on the goal (e.g. open goal angle $< \beta$), it will evaluate the positions of teammates, taking into consideration the size of their open-goal-angles, their proximity to opponents, and their ability to receive a pass. If multiple teammates satisfy these criteria, the ball is passed to the teammate with the largest open-goal-angle. As before, α and β are optimized using CMA-ES.

3.3 SARSA Agent

State-Action-Reward-State-Action (SARSA) is an algorithm for model-free on-policy Reinforcement Learning [22]. To train this agent, we model Half Field Offense as an episodic MDP in which a reward of $+1$ is given to the SARSA offensive agent for scoring a goal, -1 for the defense capturing the ball (or OOB/OOT), and 0 for all other timesteps. We use four high-level state features to train the agent: distance to goal, angle to goal, open goal angle, and distance to nearest opponent (if present). Because these state features are continuous, tile coding [22] is used to discretize the state space. Experiences collected by playing the game are then used to bootstrap a value function.

Similar to single agent learning, ad hoc teamwork can also be modeled as a reinforcement learning problem. The only change is in the features used. The rewards, actions, and states remain the same. Along with the four features used in single-agent SARSA, we now accommodate passing by using additional features, viz. each teammate's open goal angle, distance to nearest opponent, pass opening angle, and distance from our agent. Tile-coding is used to discretize this larger, augmented state space. SARSA updates are applied only when an episode terminates or the SARSA-agent is in possession of the ball.

4. HFO BENCHMARK RESULTS

One of the main contributions of this paper, in addition to the benchmark domain itself, is a set of initial results against which future users of the domain can benchmark their agents. Table 1 presents benchmark results for single-agent learning, multiagent learning, and ad hoc teamwork scenarios. Each of the presented results is averaged over 1000 evaluation episodes. Additionally, learning agent re-

	Scenario	Difficulty	Helios	Random High-Lv	Hand-coded High-Lv	SARSA High-Lv
Single	Offense (**1**v0)	Easy	**96.2** (72)	41.0 (186.6)	95.6 (48.9)	91.1 (61.3)
	Offense (**1**v1)	Medium	73.8 (79.1)	1.8 (242.7)	64.7 (68.7)	**88.9** (85.3)
	Offense (**1**v2)	Hard	34.1 (103.9)	.1 (206)	39.7 (71.6)	**40.4** (93.5)
	Defense (1v**1**)	Medium	**73.8** (79.1)	96.7 (73)	84.1 (54.9)	94.7 (73.5)
	Defense (2v**1**)	Hard	**81.4** (73.1)	96.9 (68.1)	84.8 (53.0)	94.1 (68.3)
Ad Hoc	Offense (**1**+1v1)	Easy	–	66.3 (93.7)	68.5 (59.9)	**91.5** (77.3)
	Offense (**1**+1 v 2)	Medium	–	24.3 (105.6)	46.4 (72)	**63.6** (92.4)
	Offense (**1**+2 v 3)	Medium	–	22.3 (103.2)	27.6 (76.4)	**34.5** (105)
	Defense (1 v 1+**1**)	Easy	–	49.6 (87.5)	46.9 (65.4)	**46.3** (86.2)
	Defense (2 v 1+**1**)	Medium	–	72.7 (76.3)	**60.1** (58.6)	64.7 (75.5)
	Defense (3 v 2+**1**)	Medium	–	58.4 (75.4)	43.0 (58)	**49.6** (74.5)
Multiagent	Offense (**2**v1)	Easy	81.4 (73.1)	.7 (361.7)	65.7 (66.1)	**92.3** (84.2)
	Offense (**2**v2)	Medium	60.0 (87.9)	0 (–)	46.7 (73.7)	**62.8** (93.8)
	Offense (**3**v3)	Medium	**38.8** (93.2)	0 (–)	25.7 (84.1)	33.1 (107.6)
	Defense (1v**2**)	Easy	**34.1** (103.9)	89.2 (83.1)	52.1 (62.9)	65.9 (80.7)
	Defense (2v**2**)	Medium	**60.0** (87.9)	91.0 (70.3)	60.5 (55.8)	77.2 (71.4)
	Defense (3v**3**)	Medium	**38.8** (93.2)	86.7 (66.5)	50.2 (56.4)	69.2 (70.3)

Table 1: HFO Benchmark Results: Each cell displays the percentage of episodes that ended with a goal (Goal Percentage) and, in parenthesis, the average number of simulated timesteps required to score a goal (Time to Goal). Examined offensive and defensive scenarios span Single-agent learning, Ad Hoc Teamwork, and Multiagent learning. Baseline results for the automated Helios teammate are omitted for Ad Hoc Teamwork, as they are identical to the Multiagent scenario. In the Scenario column, bold font indicates learning agents. Everywhere else, it identifies the agent with the best performance in that scenario.

sults (Hand-coded and SARSA) are averages over ten independent training runs. The benchmark agents are open-source and publicly available as a part of the HFO repository, enabling the results in Table 1 to be easily reproduced. Results from each of the different learning paradigms are discussed in greater detail below.

4.1 Single-Agent Learning

We examine three single agent HFO tasks: Scoring on an empty goal, Scoring on a keeper, and Protecting the goal. Even against an empty goal none of the agents scores every time. This is because the RoboCup 2D simulator adds noise to the perceptions and actions taken by the agents, resulting in occasionally missed shots. Once a keeper is added, the scoring percentage of all offensive agents drops drastically for all agents except SARSA. Likewise, the average number of steps required to score a goal increases by 22, indicating the sharp difficulty increase between these two tasks. However, it is no easier to play Keeper. As the results indicate, an offensive Helios-agent is just as effective scoring on an empty goal as it is against the random agent.

4.2 Ad Hoc Teamwork

Ad Hoc teamwork scenarios require the learning agent to cooperate with a Helios-controlled teammate without the benefit of pre-coordination. Playing with the Helios teammate, the random agent receives a substantial boost of 24.7 goal percentage points (GPPs) on offense and deters 28.7 more GPPs when on defense. On the other hand, learning agents have more trouble adapting their play styles to the unknown teammate. As a result, Hand-coded and SARSA agents experience less improvement, as discussed in the next section.

4.3 Multiagent Learning

Multiagent learning involves multiple agents learning in each others' presence. A learning teammate presents an opportunity for improved performance since the two agents' strategies can co-adapt. However, a non-stationary teammate policy can also be a liability if a new behavior violates existing cooperative strategies.

In order to analyze the quality of Helios versus learning teammates, we examine differences between the Ad Hoc and multiagent scenarios: The Hand-coded agent shows a slight performance increase when paired with a Helios teammate instead of a Hand-coded teammate: an average improvement of 1.5 GPPs on offense and 4.3 GPPs on defense. When paired with Helios, SARSA's goal percentage increases by only .5 GPPs on offense and a substantial 17.2 GPP reduction on defense. These trends suggests that for SARSA and Hand-coded agents, having a highly competent teammate is more valuable than a learning teammate.

4.4 Analysis

Due to the many different scenarios supported by the HFO environment, it is desirable to quantify and compare the aggregate performance of different agents. To accomplish this task we recommend a one-way Analysis of Variance (ANOVA) test. Shown in Figure 4, this test shows the aggregate offensive and defensive capabilities of the learning agents as well as the Helios expert. On offense, a clear hierarchy emerges with SARSA learning agents outperforming Hand-coded agents, who in turn perform better than the random agent. On defense, the order is different, with Helios and Hand-coded agents preventing goals more effectively than SARSA and random agents.

Overall, the high performance of the Helios policy indicates that expert hand-coded approaches are very strong.

We expect that future learning agents will be able to outperform Helios agents, perhaps by learning better skills in the low-level action space.

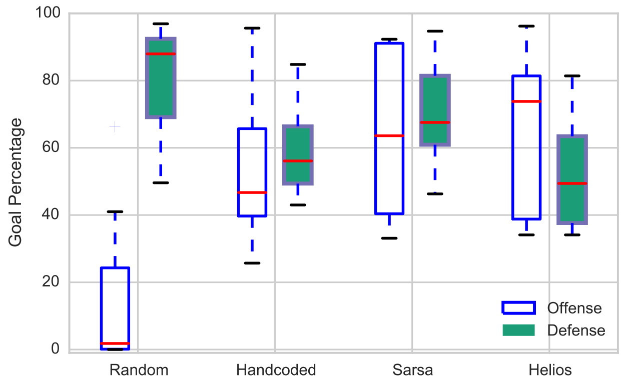

Figure 4: One-way Analysis of Variance (ANOVA) shows aggregate offensive (hollow) and defensive (filled) capabilities of each agent across all scenarios examined in Table 1: Offensively, Hand-coded, Sarsa, and Helios agents statistically significantly outperform the random agent ($p < .01$). On defense, Helios is signficiantly better than random and Sarsa agents ($p < .01$), but not significantly better than hand-coded.

5. OPEN CHALLENGES

The HFO Environment includes tasks that range in difficulty from easy to hard. Easy tasks feature favorable conditions for the learning agents - conditions such as teammates outnumbering opponents. Additionally, we outline several hard learning problems in which the opponents outnumber teammates. Two single-agent examples are 1) playing offense against two defenders and 2) playing keeper against two attackers. Both of these scenarios are yet unsolved by the current algorithms and offer much room for improvement. Using the HFO Environment it is trivial to create even harder scenarios where the odds are further stacked against the learning agent. Addressing such scenarios may be a key to discovering new strategies for competitive RoboCup agents.

A second open challenge is learning in the low-level state action space. Learning in this space is complicated by the necessity of dealing with continuous actions, and the low-level random agent demonstrates that acting randomly is insufficient to score even a single goal. However, acting in this space offers the most fine-grained control over agent's behavior and may be the key to discovering novel skills and strategies.

A final challenge is to use the inter-agent communication facilities provided by HFO to aid in coordinated tasks. The existing benchmark agents learn to cooperate without communication, but could plausibly benefit from learning to communicate as they learn to perform the cooperative task of scoring or defending goals.

We hope these challenges will spark interest in the community as much as they do in the authors. We now examine related work and conclude.

6. RELATED WORK

Progress in machine learning is driven both by the development of new algorithms and the availability of high-quality, publicly-accessible training data. Examples of highly influential supervised and unsupervised datasets include Fisher's seminal Iris dataset [9], the UCI Machine Learning repository [4], the Penn Treebank [16], MNIST handwritten digit recognition [14], ImageNet large scale visual recognition challenge [18], and Pascal Visual Object Classes [8].

Instead of datasets, reinforcement learning is driven by challenging domains. Examples of influential domains include classics like grid-world, mountain-car, and cart-pole [22], as well as more recent additions such as octopus-arm, Tetris and Pac-Man. Reinforcement learning competitions [27] featuring these domains drive development of new algorithms and agents.

One of the main inspirations for this paper is the Arcade Learning Environment [7], which has helped advance the field of AI by providing a testbed and evaluation methodology for general game playing algorithms. The HFO environment provides a similar platform, with less emphasis on generality and more emphasis on cooperation and multiagent learning.

7. CONCLUSION

It has been ten years since Stone et al. presented the RoboCup 2D Keepaway domain [21]. Keepaway created interest, sparked research, and has served for many years as a testbed for new AI algorithms [19, 25, 26]. The HFO environment is similar in spirit, but features an expanded range of tasks spanning a spectrum of difficulty levels. A high-level discrete action space allows agents to learn quickly by harnessing the same behaviors as the expert Helios agent, while a low-level continuous state action space enables researchers to investigate cutting-edge techniques for reinforcement learning in parameterized-continuous spaces featuring partial observability and opportunities for multiagent coordination.

More than just state and actions spaces, the HFO Environment features the capability to explore single agent learning, Ad Hoc teamwork, Multiagent learning, and imitation learning. In this paper, we presented benchmark results demonstrating the capabilities of reinforcement learning and hand-coded agents in each of these tasks. Furthermore, we presented an evaluation methodology and strategy for quantifying aggregate agent performance and identified several open research challenges. Using these techniques, we expect the community will be able to quickly develop, interface, and evaluate novel agents that will advance the state of the art in multiagent learning and ad hoc teamwork.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (CNS-1330072, CNS-1305287), ONR (21C184-01), AFRL (FA8750-14-1-0070), AFOSR (FA9550-14-1-0087), and Yujin Robot.

REFERENCES

[1] The robocup soccer simulator. `http://sourceforge.net/projects/sserver/`. Accessed: 2016-02-01.

[2] Hidehisa Akiyama. Agent2d base code. `https://osdn.jp/projects/rctools/`, 2010.

[3] Brenna Argall, Sonia Chernova, Manuela M. Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[4] Arthur Asuncion and David J Newman. Uci machine learning repository. `http://archive.ics.uci.edu/ml/`, 2007.

[5] Samuel Barrett. *Making Friends on the Fly: Advances in Ad Hoc Teamwork*. PhD thesis, The University of Texas at Austin, Austin, Texas, USA, December 2014.

[6] Samuel Barrett and Peter Stone. Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2010–2016, January 2015.

[7] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[8] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[9] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188, 1936.

[10] Nikolaus Hansen. The cma evolution strategy: A comparing review. In *Towards a New Evolutionary Computation*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer Berlin Heidelberg, 2006.

[11] Junling Hu and Michael P. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[12] Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone. Half field offense in robocup soccer: A multiagent reinforcement learning case study. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico G. Sorrenti, and Tomoichi Takahashi, editors, *RoboCup*, volume 4434 of *Lecture Notes in Computer Science*, pages 72–85. Springer, 2006.

[13] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Agents*, pages 340–347, 1997.

[14] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. `http://yann.lecun.com/exdb/mnist/`, 2010.

[15] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, July 10-13, 1994*, pages 157–163. Morgan Kaufmann, 1994.

[16] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993.

[17] Warwick Masson and George Konidaris. Reinforcement learning with parameterized actions. *CoRR*, abs/1509.01644, 2015.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[19] Vishal Soni and Satinder Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *AAAI*, volume 6, pages 494–499, 2006.

[20] Peter Stone, Gal A. Kaminka, Sarit Kraus, and Jeffrey S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*, July 2010.

[21] Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup soccer keepaway. *Adaptive Behaviour*, 13(3):165–188, 2005.

[22] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[23] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[24] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Michael N. Huhns and Munindar P. Singh, editors, *Readings in Agents*, pages 487–494. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[25] Matthew E Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

[26] Phillip Verbancsics and Kenneth O Stanley. Evolving static representations for task transfer. *The Journal of Machine Learning Research*, 11:1737–1769, 2010.

[27] Shimon Whiteson, Brian Tanner, and Adam White. The reinforcement learning competitions. *AI Magazine*, 31(2):81–94, 2010.