In *Autonomous Agents and Multiagent Systems, AAMAS 2017 Workshops, Best Papers*, Lecture Notes in Artificial Intelligence,
pp. 168--186, Springer International Publishing, 2017.

Evaluating Ad Hoc Teamwork Performance in Drop-In Player Challenges

Patrick MacAlpine and Peter Stone

Department of Computer Science, The University of Texas at Austin, USA
{patmac,pstone}@cs.utexas.edu

Abstract. Ad hoc teamwork has been introduced as a general challenge for AI and especially multiagent systems [16]. The goal is to enable autonomous agents to band together with previously unknown teammates towards a common goal: collaboration without pre-coordination. A long-term vision for ad hoc teamwork is to enable robots or other autonomous agents to exhibit the sort of flexibility and adaptability on complex tasks that people do, for example when they play games of "pick-up" basketball or soccer. As a testbed for ad hoc teamwork, autonomous robots have played in pick-up soccer games, called "drop-in player challenges", at the international RoboCup competition. An open question is how best to evaluate ad hoc teamwork performance—how well agents are able to coordinate and collaborate with unknown teammates—of agents with different skill levels and abilities competing in drop-in player challenges. This paper presents new metrics for assessing ad hoc teamwork performance, specifically attempting to isolate an agent's coordination and teamwork from its skill level, during drop-in player challenges. Additionally, the paper considers how to account for only a relatively small number of pick-up games being played when evaluating drop-in player challenge participants.

Keywords: Ad Hoc Teams, Multiagent Systems, Teamwork, Robotics

1 Introduction

The increasing capabilities of robots and their decreasing costs is leading to increased numbers of robots acting in the world. As the number of robots grows, so will their need to cooperate with each other to accomplish shared tasks. Therefore, a significant amount of research has focused on multiagent teams. However, most existing techniques are inapplicable when the robots do not share a coordination protocol, a case that becomes more likely as the number of companies and research labs producing these robots grows. To deal with this variety of previously unseen teammates, robots can reason about *ad hoc teamwork* [16]. When participating as part of an ad hoc team, agents need to cooperate with previously unknown teammates in order to accomplish a shared goal. Reasoning about these settings allows robots to be robust to the teammates they may encounter.

In [16], Stone et al. argue that ad hoc teamwork is "ultimately an empirical challenge." Therefore, a series of "drop-in player challenges" [14, 5, 6] have been held at the RoboCup competition,[1] a well established multi-robot competition. These challenges bring together real and simulated robots from teams from around the world to investigate the current ability of robots to cooperate with a variety of unknown teammates.

In each game of the challenges, robots are drawn from the participating teams and combined to form a new team. These robots are not informed of the identities of any of their teammates, but they are able to share a small amount of information using a limited standard communication protocol that is published in advance. These robots then have to quickly adapt to their teammates over the course of a single game and discover how to intelligently share the ball and select which roles to play.

Currently in drop-in player challenges, a metric used to evaluate participants is the average goal difference received by an agent across all games that an agent plays in. An agent's average goal difference is strongly correlated with how skilled an agent is, however, and is not necessarily a good way of evaluating an agent's *ad hoc teamwork performance*—how well agents are able to coordinate and collaborate with unknown teammates. Additionally, who an agent's teammates and opponents are during a particular drop-in player game strongly affects the game's result, and it may not be feasible to play enough games containing all possible combinations of agents on different ad hoc teams, thus the agent assignments to the ad hoc teams of the games that are played may bias an agent's average goal difference.

This paper presents new metrics for assessing ad hoc teamwork performance, specifically attempting to isolate an agent's coordination and teamwork from its skill level, during drop-in player challenges. Additionally, the paper considers how to account for only a relatively small number of games being played when evaluating drop-in player challenge participants.

The rest of the paper is structured as follows. A description of the the RoboCup 3D simulation domain used for this research is provided in Section 2. Section 3 explains the drop-in player challenge. Section 4 details our metric for evaluating ad hoc teamwork performance, and analysis of this metric is provided in Section 5. Section 6 discusses an extension to this metric when one can add agents with different skill levels, but the same level of teamwork, to a drop-in player challenge. How to account for a limited number of drop-in player games being played when evaluating ad hoc teamwork performance is presented in Section 7. A case study of the 2015 RoboCup 3D simulation drop-in player challenge demonstrating our work is analyzed in Section 8. Section 9 situates this work in literature, and Section 10 concludes.

[1] `http://www.robocup.org/`

2 RoboCup Domain Description

Robot soccer has served as an excellent research domain for autonomous agents and multiagent systems over the past decade and a half. In this domain, teams of autonomous robots compete with each other in a complex, real-time, noisy and dynamic environment, in a setting that is both collaborative and adversarial. RoboCup includes several different leagues, each emphasizing different research challenges. For example, the humanoid robot league emphasizes hardware development and low-level skills, while the 2D simulation league emphasizes more high-level team strategy. In all cases, the agents are all fully autonomous.

The RoboCup 3D simulation environment—the setting for our work—is based on SimSpark,[2] a generic physical multiagent systems simulator. SimSpark uses the Open Dynamics Engine[3] (ODE) library for its realistic simulation of rigid body dynamics with collision detection and friction. ODE also provides support for the modeling of advanced motorized hinge joints used in the humanoid agents.

The robot agents in the simulation are homogeneous and are modeled after the Aldebaran Nao robot. The agents interact with the simulator by sending torque commands and receiving perceptual information. Each robot has 22 degrees of freedom, each equipped with a perceptor and an effector. Joint perceptors provide the agent with noise-free angular measurements every simulation cycle (20 ms), while joint effectors allow the agent to specify the torque and direction in which to move a joint. Although there is no intentional noise in actuation, there is slight actuation noise that results from approximations in the physics engine and the need to constrain computations to be performed in real-time. Abstract visual information about the environment is given to an agent every third simulation cycle (60 ms) through noisy measurements of the distance and angle to objects within a restricted vision cone (120°). Agents are also outfitted with noisy accelerometer and gyroscope perceptors, as well as force resistance perceptors on the sole of each foot. Additionally, agents can communicate with each other every other simulation cycle (40 ms) by sending 20 byte messages.

Games consist of two 5 minute halves of 11 versus 11 agents on a field size of 20 meters in width by 30 meters in length. Figure 1 shows a visualization of the simulated robot and the soccer field during a game.

3 Drop-In Player Challenge

For RoboCup 3D drop-in player challenges[4] each participating team contributes two drop-in field players to a game. Each drop-in player competes in full 10

[2] http://simspark.sourceforge.net/
[3] http://www.ode.org/
[4] Full rules of the challenges can be found at http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/2015_dropin_challenge/

Fig. 1: A screenshot of the Nao-based humanoid robot (left), and a view of the soccer field during a 11 versus 11 game (right).

minute games (two 5 minute halves) with both teammates and opponents consisting of other drop-in field players. No goalies are used during the challenge to increase the probability of goals being scored.

Ad hoc teams are chosen by a greedy algorithm given in Algorithm 1 that attempts to even out the number of times agents from different participants in a challenge play with and against each other. In lines 6 and 7 of the algorithm agents are iteratively added to teams by `getNextAgent()` which uses the following ordered preferences to select agents that have:

1. Played fewer games.
2. Played against fewer of the opponents.
3. Played with fewer of the teammates.
4. Played a lower maximum number of games against any one opponent or with any one teammate.
5. Played a lower maximum number of games against any one opponent.
6. Played a lower maximum number of games with any one teammate.
7. Random.

Algorithm 1 terminates when all agents have played at least one game with and against all other agents.

Each drop-in player can communicate with its teammates using a simple protocol, — the use of the protocol is purely optional. The protocol communicates the following information:

- player's team
- player's uniform number
- player's current (x,y) position on the field
- (x,y) position of the ball
- time ball was last seen
- if player is currently fallen over

Algorithm 1 Drop-In Team Agent Selection

Input: *Agents*

$games = \varnothing$
while not `allAgentsHavePlayedWithAndAgainstEachOther()` **do**
  $team1 := \varnothing$
  $team2 := \varnothing$
  for $i := 1$ **to** $AGENTS\_PER\_TEAM$ **do**
    $team1 \leftarrow$ `getNextAgent`$(Agents \setminus \{team1 \cup team2\})$
    $team2 \leftarrow$ `getNextAgent`$(Agents \setminus \{team1 \cup team2\})$
  $games \leftarrow \{team1, team2\}$
return *games*

A C++ implementation of the protocol is provided to all participants.

All normal game rules apply in this challenge. Each player is randomly assigned a uniform number from 2-11 at the start of a game. The challenge is scored by the average goal difference received by an agent across all games that an agent plays in.

4 Ad Hoc Teamwork Performance Metric

Since 2013 drop-in player challenges have been held at RoboCup in multiple robot soccer leagues including 3D simulation, 2D simulation, and the physical Nao robot Standard Platform League (SPL) [14, 13, 15, 5, 6]. Across these challenges there has been a high correlation between how well a team does in the challenge and how well a team performs in the main soccer competition. This correlation suggests it may be the case that better individual skills and ability—as opposed to teamwork—is a dominating factor when using average goal difference to rank challenge participants.

As drop-in player challenges are designed as a test bed for ad hoc teamwork, and the ability of an agent to interact with teammates without pre-coordination, ideally we would like to evaluate *ad hoc teamwork performance*—how well agents are able to coordinate and collaborate with unknown teammates. To measure this performance we need a way of isolating agents' ad hoc teamwork from their skill levels.

One way to infer an agent's skill level, relative to another agent, is to evaluate how agents perform in a drop-in player challenge when playing games with teams consisting entirely of their own agent. By playing two different agent teams against each other, and with each teams' members being of the same agent, we are able to directly measure the relative performance difference between the two agents. Although agents' skill levels may not be the only factor in the difference in performance between two teams—factors such as team coordination dynamics may affect performance as well—the teams' relative performance is used as a proxy for individual skills of its members. For agent team a playing agent team b we denote their skill difference, measured as the expected number of goals scored by agent team a minus the expected number of goals scored by agent team b, to be $\texttt{relSkill}(a,b)$.

Given the `relSkill` value for all agent pairs, which can be measured by having all agents play each other in a round robin style tournament, we can estimate the expected goal difference of any mixed agent team drop-in player game by summing and then averaging the `relSkill` values of all agent pairs on opposing teams. Equation 1 shows the estimated score between two mixed agent teams A and B.

$$\texttt{score}(A,B) = \frac{1}{|A||B|}\sum_{a\in A, b\in B}\texttt{relSkill}(a,b) \tag{1}$$

Next, to determine the overall skill of an agent relative to all other agents, we compute the average goal difference across all possible $\left(\binom{N}{K} * \binom{N-K}{K}\right)/2$ drop-in player mixed team game permutations for an agent, where N is the total number of agents and K is the number of agents per team, using the estimated goal difference of each game from Equation 1. We denote this value measuring the average goal difference (AGD) across all games for agent a as $\texttt{skillAGD}(a)$. Instead of explicitly computing the score for all game permutations, we can simplify computation as shown in the following example to compute $\texttt{skillAGD}(a)$ for a drop-in player challenge with agents $\{a,b,c,d\}$ and two agents on each team.

First determine the `score` of all drop-in game permutations involving agent a (`rS` used as shorthand for `relSkill`):

$$\texttt{score}(\{a,b\},\{c,d\}) = \frac{\texttt{rS}(a,c)+\texttt{rS}(a,d)+\texttt{rS}(b,c)+\texttt{rS}(b,d)}{4}$$

$$\texttt{score}(\{a,c\},\{b,d\}) = \frac{\texttt{rS}(a,b)+\texttt{rS}(a,d)+\texttt{rS}(c,b)+\texttt{rS}(c,d)}{4}$$

$$\texttt{score}(\{a,d\},\{b,c\}) = \frac{\texttt{rS}(a,b)+\texttt{rS}(a,c)+\texttt{rS}(d,b)+\texttt{rS}(d,c)}{4}$$

Averaging all scores to get $\texttt{skillAGD}(a)$, and as

$$\texttt{rS}(a,b) = -\texttt{rS}(b,a),$$

this simplifies to

$$\texttt{skillAGD}(a) = \frac{\texttt{rS}(a,b)+\texttt{rS}(a,c)+\texttt{rS}(a,d)}{6}.$$

Based on `relSkill` values canceling each other out when averaging over all drop-in game permutations, as shown in the above example, Equation 2 provides a simplified form for estimating an agent's skill.

$$\texttt{skillAGD}(a) = \frac{1}{K(N-1)}\sum_{b\in Agents\setminus a}\texttt{relSkill}(a,b) \tag{2}$$

To evaluate agents' ad hoc teamwork we also need a measure of how well they do when playing in mixed team drop-in player games. Let $\texttt{dropinAGD}(a)$ be the actual, measured average goal difference for agent a across all mixed team permutations of drop-in player games. Given an agent's `skillAGD`—estimated indirectly from `relSkill` values—and `dropinAGD`—measured directly—values, we compute a metric `teamworkAGD` for measuring an agent's teamwork. An agent's `teamworkAGD` value is computed by subtracting an agent's skill from it's measured performance in drop-in player games as shown in Equation 3.

$$\texttt{teamworkAGD}(a) = \texttt{dropinAGD}(a) - \texttt{skillAGD}(a) \quad (3)$$

The `teamworkAGD` value serves to help remove the bias of an agent's skill from its measured averaged goal difference during drop-in player challenges, and in doing so provides a metric to isolate ad hoc teamwork performance.

5 Ad Hoc Teamwork Performance Metric Evaluation

To evaluate the `teamworkAGD` ad hoc teamwork performance metric presented in Section 4, we need to be able to create agents with different known skill levels and teamwork such that an agent's skill level is independent of its teamwork. Once we have agents with known differences in skill level and teamwork relative to each other, it is possible to check if the `teamworkAGD` metric is able to isolate agents' ad hoc teamwork from their skill levels during a drop-in player challenge. For our analysis, we designed a RoboCup 3D simulation drop-in player challenge with ten agents each having one of five skill levels and either poor or non-poor teamwork—there is a single agent for every combination of skill level and teamwork type—as follows.

We first created five drop-in player agents with different skill levels determined by how fast an agent is allowed to walk—the maximum walking speed is the only difference between the agents. While walking speed is only one factor for determining an agent's skill level—other factors such as how far an agent can kick the ball and how fast it can get up after falling are important too—by varying their maximum walking speed we ensure agents' overall skill levels differ significantly. The five agents, from highest to lowest skill level, were allowed to walk up to the following maximum walking speeds: 100%, 90%, 80%, 70%, 60%. We then played a round robin tournament with each of the five agents playing 100 games against each other. During these games members of each team consisted of all the same agent. Results from these games of the `relSkill` values of agents with different skill levels are shown in Table 1.

From the values in Table 1 we then compute the agents' skills relative to each other (`skillAGD`) using Equation 2. When doing so we model the drop-in player challenge as being between ten participants consisting of two agents from each of the five skill levels. We also assume that the average goal difference between two agents of the same skill level is 0.[5] Agents' skill values are shown in Table 2.

[5] Empirically we have found that the average goal difference when one team plays itself approaches 0 across many games.

Table 1: Average goal difference of agents with different skill levels when playing 100 games against each other. A positive goal difference means that the row agent is winning. The number at the end of the agents' names refers to their maximum walk speed percentages.

	Agent60	Agent70	Agent80	Agent90
Agent100	1.73	1.36	0.78	0.24
Agent90	1.32	0.94	0.45	
Agent80	0.71	0.52		
Agent70	0.16			

Table 2: Skill values (`skillAGD`) for agents with different skill levels. The number at the end of the agents' names refers to their maximum walk speed percentages.

Agent	`skillAGD`
Agent100	0.183
Agent90	0.110
Agent80	0.000
Agent70	-0.118
Agent60	-0.174

The default strategy for each of our drop-in player agents is for an agent to go to the ball if it is the closest member of its team to the ball. Once at the ball, an agent then attempts to kick or dribble the ball toward the opponent's goal. If the agent is not the closest to the ball, it waits at a position two meters behind the ball in a supporting position.

To create agents with poor teamwork, we made modified versions of each of the five different skill level agents such that the modified versions will still go to the ball if an unknown teammate—an agent that is not the exact same type—is closer or even already at the ball. These modified agents, which we refer to as "PT agents" for poor teamwork, can interfere with their unknown teammates and impede progress of the team as a whole. The only teammates they will not interfere with are known agent teammates—agents of the same type with the same maximum walking speed and poor teamwork attribute.

We played a drop-in player challenge with all ten agent types. The total number of possible drop-in team combinations is $(\binom{10}{5} * \binom{5}{5})/2 = 126$. Each combination was played ten times, resulting in a total of 1260 games. Data from these games showing each agent's `dropinAGD`, as well as the agents' `skillAGD` and computed `teamworkAGD`, are shown in Table 3. Note that a poor teamwork agent has the same `skillAGD` as the non-poor teamwork agents with the same walking speed—both agents behave identically when playing on a team consisting of all their own agents.

While the data in Table 3 shows a direct correlation of agents with higher skill levels having higher `dropinAGD` values, the `teamworkAGD` values rank all normal

Table 3: Skill value, drop-in player tournament average goal difference, and ad hoc teamwork performance metric for different agents sorted by `teamworkAGD`.

Agent	`skillAGD`	`dropinAGD`	`teamworkAGD`
Agent70	-0.118	0.017	0.135
Agent60	-0.174	-0.055	0.119
Agent80	0.000	0.087	0.087
Agent100	0.183	0.204	0.021
Agent90	0.110	0.123	0.013
PTAgent60	-0.174	-0.196	-0.022
PTAgent70	-0.118	-0.169	-0.051
PTAgent100	0.183	0.109	-0.074
PTAgent80	0.000	-0.101	-0.101
PTAgent90	0.110	-0.018	-0.128

agents above poor teamwork agents. As `teamworkAGD` is able to discern between agents with different levels of teamwork, despite the agents having different levels of skill, `teamworkAGD` is a viable metric for analyzing ad hoc teamwork performance. However, there is a trend for agents with lower `skillAGD` values to have higher `teamworkAGD` values. We discuss and account for this trend in the next section.

6 Normalized Ad Hoc Teamwork Performance Metric

Part of the reason `teamworkAGD` in Table 3 is able to separate the agents with poor teamwork independent of an agent's skill level is due to agents with the same teamwork having similar values of `teamworkAGD`. Empirically we have noticed that is not always the case that teams with the same teamwork have similar `teamworkAGD` values. When skill levels between agents are more spread out, there is a trend for agents with lower skill levels to have higher values for `teamworkAGD`. This trend can be seen in Table 4 containing data from a drop-in player challenge with agents having maximum walking speeds between 100% and 40% of the possible maximum walking speed.

With the trend of agents with lower `skillAGD` having higher values for `teamworkAGD`, the poor teamwork PTAgent50 agent in Table 4 has a higher `teamworkAGD` than several of the non-poor teamwork agents.

To account for agents with the same teamwork, but different skill levels, we can normalize these agents' `teamworkAGD` values to 0. We define the value added to each of these agents' `teamworkAGD` values to set them to 0 as the agents' `normOffset` values. Thus for a set of multiple agents A with the same teamwork, and for every agent $a \in A$, we let $\texttt{normOffset}(a) = -\texttt{teamworkAGD}(a)$. This normalization produces a `normTeamworkAGD` value as shown in Equation 4.

$$\texttt{normTeamworkAGD}(a) = \texttt{teamworkAGD}(a) + \texttt{normOffset}(a) \qquad (4)$$

Table 4: Skill value, drop-in player tournament average goal difference, and ad hoc teamwork performance metric for different agents sorted by `teamworkAGD`.

Agent	`skillAGD`	`dropinAGD`	`teamworkAGD`
Agent40	-0.710	-0.270	0.440
Agent50	-0.226	-0.129	0.097
Agent55	-0.142	-0.081	0.061
Agent100	0.412	0.416	0.004
PTAgent50	-0.226	-0.230	-0.004
Agent90	0.296	0.259	-0.037
Agent70	0.028	-0.005	-0.033
Agent85	0.245	0.176	-0.069
PTAgent70	0.028	-0.179	-0.207
PTAgent90	0.296	0.043	-0.253

While `normTeamworkAGD` will give the same value of 0 for agents that we know to have the same teamwork, we want to estimate `normOffset`, and then compute `normTeamworkAGD`, for agents that we do not necessarily know about their teamwork. To estimate `normOffset` values we first plot the `normOffset` values relative to `teamworkAGD` values for the agents with the same teamwork, and then fit a curve through these points. To intersect each point, we do a least squares fit to a $n-1$ degree polynomial, where n is the number of points we are fitting the curve to. Then, to estimate any agent's `normOffset` value, we choose the point on this curve corresponding to the agent's `skillAGD`. A curve generated by the `normOffset` values normalizing `teamworkAGD` to 0 for Agent100, Agent85, Agent70, Agent55, and Agent40 from Table 4 is shown in Figure 2.

Table 5 shows `normOffset` and `normTeamworkAGD` values for the agents in Table 4. The `normOffset` values for agents with 50% and 90% speeds are estimated. Considering that `normTeamworkAGD` is able to discern between agents with different levels of teamwork, it is a useful metric for analyzing ad hoc teamwork performance when agents with the same teamwork have larger differences in their `teamworkAGD` values. To compute `normTeamworkAGD`, however, a set of agents with the same teamwork, but different skill levels, must be included in a drop-in player challenge.

7 Drop-In Player Game Prediction

Computing `dropinAGD` requires results from all possible agent to team assignment permutations of drop-in player games. The number of games grows factorially as this is $\left(\binom{N}{K} * \binom{N-K}{K}\right)/2$ drop-in player games, where N is the total number of agents and K is the number of agents per team. Playing all permutations of drop-in player games may not be tractable or feasible, especially when drop-in player competitions involve physical robots [5, 6].

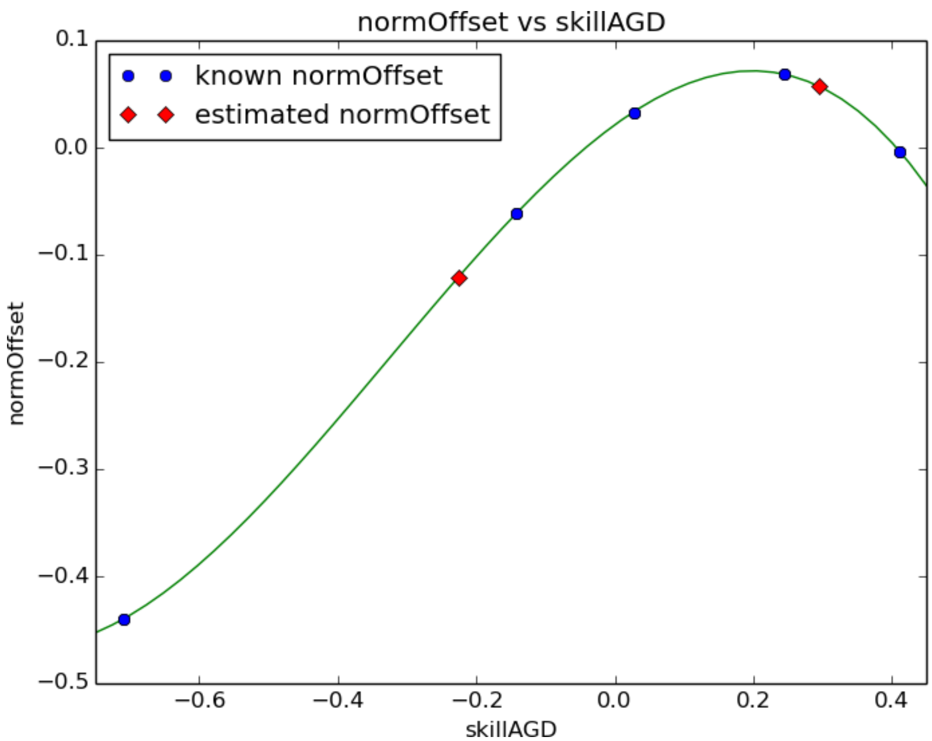

Fig. 2: Curve of `normOffset` vs `skillAGD` based on `normOffset` values normalizing `teamworkAGD` to 0 for Agent100, Agent85, Agent70, Agent55, and Agent40 from Table 4. Both data points used to generate the curve (blue dots) and points used to estimate `normOffset` for agents walking at 50% and 90% speeds (red diamonds) are shown.

Table 5: `teamworkAGD`, `normOffset`, and `normTeamworkAGD` values for the agents in Table 4 sorted by `normTeamworkAGD`.

Agent	`teamworkAGD`	`normOffset`	`normTeamworkAGD`
Agent90	-0.037	0.057	0.020
Agent55	0.061	-0.061	0.000
Agent40	0.440	-0.440	0.000
Agent100	0.004	-0.004	0.000
Agent70	-0.033	0.033	0.000
Agent85	-0.069	0.069	0.000
Agent50	0.097	-0.121	-0.024
PTAgent50	-0.004	-0.121	-0.125
PTAgent70	-0.207	0.033	-0.174
PTAgent90	-0.253	0.057	-0.196

To account for fewer numbers of drop-in player games being played, a prediction model can be built, based on data from previously played drop-in player games, to predict the scores of games that have not been played. Combining data from both the scores of games played and predicted games then allows for `dropinAGD` to be estimated.

One way to predict the scores of drop-in player games is to model them as a linear system of equations. More specifically, we can represent a drop-in player game as a linear equation with strength coefficients for individual agents, cooperative teammate coefficients for pairs of agents on the same team, and adversarial opponent coefficients for pairs of agents on opposing teams.

Given two drop-in player teams A and B, $\texttt{score}(A, B)$ is modeled as the sum of strength coefficients S,

$$\sum_{a \in Agents} S_a * \begin{cases} 1 & \text{if } a \in A \\ -1 & \text{if } a \in B \\ 0 & \text{otherwise} \end{cases}$$

teammate coefficients T,

$$\sum_{a \in Agents, b \in Agents, a<b} T_{a,b} * \begin{cases} 1 & \text{if } a \in A \text{ and } b \in A \\ -1 & \text{if } a \in B \text{ and } b \in B \\ 0 & \text{otherwise} \end{cases}$$

and opponent coefficients O,

$$\sum_{a \in Agents, b \in Agents, a<b} O_{a,b} * \begin{cases} 1 & \text{if } a \in A \text{ and } b \in B \\ -1 & \text{if } a \in B \text{ and } b \in A. \\ 0 & \text{otherwise} \end{cases}$$

There are N strength coefficients, and $\binom{N}{2}$ of both teammate and opponent coefficients, for a total of $N + 2\binom{N}{2}$ coefficients.

To solve for the coefficients in the system of linear equations least squares regression is used. There needs to be enough data from games such that every agent has played with and against every other agent, however, so that there is at least one instance of every coefficient being multiplied by a non-zero number. Using Algorithm 1, with 10 agents total and 5 agents per team, having every coefficient multiplied by a non-zero number requires only 5 games. Figure 3 shows how the number of games required to create a prediction model increases as the number of agents increase when using Algorithm 1. Although it is possible to create a prediction model with a minimum number of games, such a system will be very underdetermined and more games will result in better predictions.

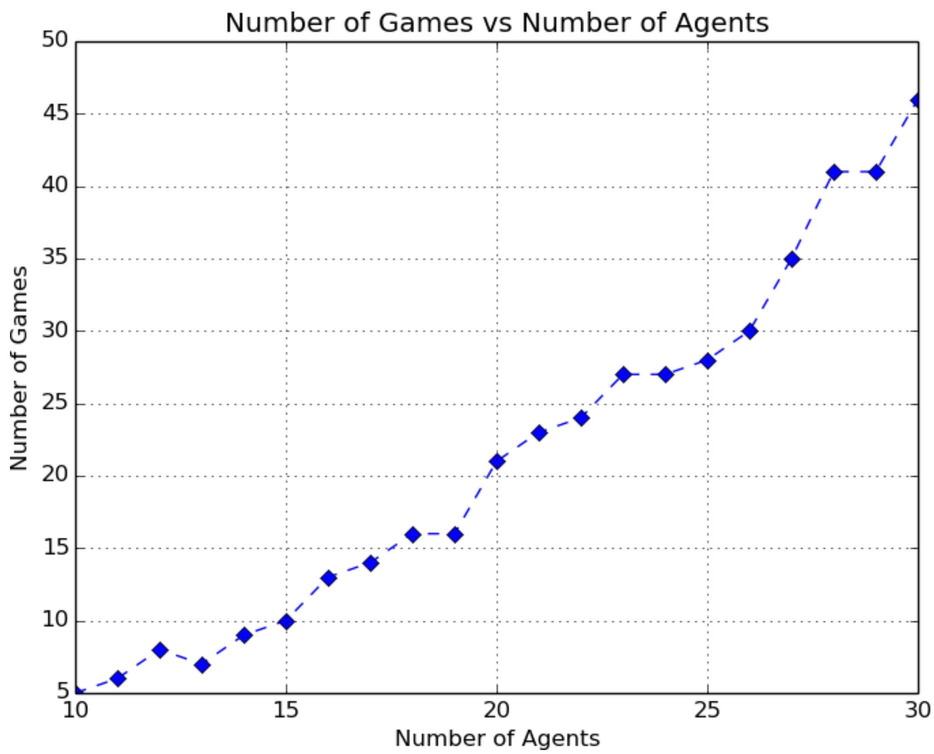

Fig. 3: The number of games required to play all agents with and against every other agent using Algorithm 1 as the number of agents increase. This data assumes there are five agents on each team.

As an example of our prediction model, Tables 6 and 7 show predicted values of `dropinADG` created from game scores generated by prediction models built from half the game data—data from 630 games—used to compute `dropinADG` values in Tables 3 and 4 respectively. More specifically, data from games encompassing half of all possible agent to team assignment permutations of drop-in player games—the first 63 out of 126 possible unique team permutations generated by letting Algorithm 1 continue to run even after all teams have played with and against each other—was used to build the prediction models.

Table 6: The `dropinAGD` values from Table 3 (computed from all 1260 games) compared to both `dropinAGD` values from half the games played used to compute the data in Table 3 ($\frac{1}{2}$ `dropinAGD` with 630 games), and predicted `dropinAGD` values generated from a prediction model built from the game data used to compute $\frac{1}{2}$ `dropinAGD` (Pred. `dropinAGD` with 630 games). The difference (error) from the true `dropinAGD` values for both half the games played and predicted `dropinAGD` are shown in parentheses.

Agent	`dropinAGD` 1260 games	$\frac{1}{2}$ `dropinAGD` 630 games	Pred. `dropinAGD` 630 games
Agent100	0.204	0.194 (0.010)	0.223 (0.019)
Agent90	0.123	0.133 (0.010)	0.122 (0.001)
PTAgent100	0.109	0.114 (0.005)	0.117 (0.008)
Agent80	0.087	0.121 (0.034)	0.095 (0.008)
Agent70	0.017	0.006 (0.011)	0.021 (0.004)
PTAgent90	-0.018	-0.022 (0.004)	-0.019 (0.001)
Agent60	-0.055	-0.105 (0.050)	-0.094 (0.039)
PTAgent80	-0.101	-0.060 (0.041)	-0.073 (0.028)
PTAgent70	-0.169	-0.194 (0.025)	-0.181 (0.012)
PTAgent60	-0.196	-0.187 (0.009)	-0.212 (0.016)

The majority of the predicted `dropinAGD` values in Tables 6 and 7 are closer to the true `dropinAGD` values than that of their counterpart $\frac{1}{2}$ `dropinAGD` values computed directly from the games used to build the prediction models. Furthermore, the predicted `dropinAGD` values reduce the mean squared error relative to the $\frac{1}{2}$ `dropinAGD` values: from 6.405×10^{-4} to 3.212×10^{-4} and from 3.076×10^{-3} to 9.068×10^{-4} for Tables 6 and 7 respectively.

8 Case Study: RoboCup 2015 Drop-in Player Challenge

Table 8 shows the results of computing `normTeamworkAGD` values for the ten released binaries of the 2015 RoboCup 3D simulation drop-in player challenge [15] participants. In doing so we added five agents with different skill levels but the same teamwork to the challenge: Agent100, Agent80, Agent65, Agent50, and Agent30. These agents, chosen specifically to have `skillAGD` values that span across the range of the 2015 RoboCup 3D simulation drop-in player challenge participants, are the same as the drop-in player agents used in our previous experiments—with the number at the end of the agents' names referring to their

Table 7: The `dropinAGD` values from Table 4 (computed from all 1260 games) compared to both `dropinAGD` values from half the games played used to compute the data in Table 4 ($\frac{1}{2}$ `dropinAGD` with 630 games), and predicted `dropinAGD` values generated from a prediction model built from the game data used to compute $\frac{1}{2}$ `dropinAGD` (Pred. `dropinAGD` with 630 games). The difference (error) from the true `dropinAGD` values for both half the games played and predicted `dropinAGD` are shown in parentheses.

Agent	`dropinAGD` 1260 games	$\frac{1}{2}$ `dropinAGD` 630 games	Pred. `dropinAGD` 630 games
Agent100	0.416	0.454 (0.038)	0.436 (0.020)
Agent90	0.259	0.356 (0.097)	0.296 (0.037)
Agent85	0.176	0.203 (0.027)	0.201 (0.025)
PTAgent90	0.043	0.105 (0.062)	0.048 (0.005)
Agent70	-0.005	-0.019 (0.014)	-0.016 (0.011)
Agent55	-0.081	-0.168 (0.087)	-0.132 (0.051)
Agent50	-0.129	-0.121 (0.008)	-0.098 (0.031)
PTAgent70	-0.179	-0.241 (0.062)	-0.173 (0.006)
PTAgent50	-0.230	-0.238 (0.008)	-0.241 (0.011)
Agent40	-0.270	-0.330 (0.060)	-0.323 (0.053)

maximum walk speed percentages—except now the agents are made slightly more competitive by having them communicate to their known teammates (those of the exact same agent type) where they are kicking the ball. Once an agent hears from a teammate the location its teammate is kicking the ball to, the agent then runs toward that location in anticipation of the ball being kicked there.

As there are 15 agents in the challenge, which would require $\left(\binom{15}{5} * \binom{10}{5}\right)/2 =$ 378,378 possible agent assignments for drop-in player games, we only played 1000 games—the first 1000 team permutations generated by letting Algorithm 1 continue to run even after all teams have played with and against each other—and then built a prediction model from the results of these games to compute predicted `dropinAGD` values for all agents. Using a prediction model is the only way for us to compute `dropinAGD`, and in turn `normTeamworkAGD`, given the large increase in the number of games needed to compute `dropinAGD` when adding five extra agents. The curve used to estimate `normOffset` values, and generated by the `normOffset` values normalizing `teamworkAGD` to 0 for Agent100, Agent80, Agent65, Agent50, and Agent30 from Table 8, is shown in Figure 4.

When analyzing the data in Table 8 we empirically find that most of the agents with lower `teamworkAGD` values interfere with their teammates when going to the ball. On the other hand, UTAustinVilla—the agent with the highest `teamworkAGD` value—purposely avoids running into teammates, and also checks to ensure it will not collide with other agents before attempting to kick the ball on its team's kickoffs [14].

Table 8: Computed values from released binaries of the 2015 RoboCup 3D simulation drop-in player challenge sorted by `normTeamworkAGD`. Values for `skillAGD` were computed from every agent playing 100 games against each of the other agents with teams consisting of all the same agent. Predicted `dropinAGD` (Pred. `dropinAGD`) values were computed using a prediction model built from the results of playing 1000 drop-in player games—only a very small partial amount of all 378,378 possible agent assignments for drop-in player games. These predicted `dropinAGD` values were then used in the computation of `teamworkAGD`, `normOffset`, and `normTeamworkAGD` values.

Agent	`skillAGD`	`dropinAGD` 1000 games	Pred. `dropinAGD` 1000 games	`teamworkAGD`	`normOffset`	`normTeamworkAGD`
UTAustinVilla	0.932	1.184	1.178	0.246	0.129	0.375
FCPortugal	0.384	0.228	0.262	-0.122	0.267	0.145
magmaOffenburg	0.038	-0.069	-0.047	-0.085	0.139	0.054
Agent100	1.095	1.004	1.031	-0.064	0.064	0
Agent80	0.772	0.586	0.577	-0.195	0.195	0
Agent65	0.355	0.085	0.091	-0.264	0.264	0
Agent50	-0.278	-0.151	-0.129	0.149	-0.149	0
Agent30	-1.456	-0.432	-0.437	1.019	-1.019	0
BahiaRT	0.328	0.044	-0.029	-0.357	0.260	-0.097
RoboCanes	0.178	-0.207	-0.199	-0.377	0.216	-0.161
FUT-K	0.520	-0.027	0.029	-0.491	0.263	-0.228
Apollo3D	-0.533	-0.486	-0.506	0.027	-0.465	-0.438
HfutEngine3D	-1.124	-0.468	-0.470	0.654	-1.100	-0.446
CIT3D	-0.574	-0.581	-0.589	-0.015	-0.519	-0.534
Nexus3D	-0.676	-0.713	-0.763	-0.087	-0.653	-0.740

9 Related Work

Multiagent teamwork is a well studied topic, with most work tackling the problem of creating standards for coordinating and communicating. One such algorithm is STEAM [17], in which team members build up a partial hierarchy of joint actions and monitor the progress of their plans. STEAM is designed to communicate selectively, reducing the amount of communication required to coordinate the team. In [7], Grosz and Kraus present a reformulation of the SharedPlans, in which agents communicate their intents and beliefs and use this information to reason about how to coordinate joint actions. In addition, SharedPlans provides a process for revising agents' intents and beliefs to adapt to changing conditions. In the TAEMS framework [9], the focus is on how the task environment affects agents and their interactions with one another. Specifically, agents reason about what information is available for updating their mental state. While these algorithms have been shown to be effective, they require that the teammates share their coordination framework.

On the other hand, ad hoc teamwork focuses on the case where the agents do not share a coordination algorithm. In [12], Liemhetcharat and Veloso reason about selecting agents to form ad hoc teams. Barrett et al. [2] empirically evaluate an MCTS-based ad hoc team agent in the pursuit domain, and Barrett and Stone [1] analyze existing research on ad hoc teams and propose one way to categorize ad hoc teamwork problems. Other approaches include Jones et al.'s work [10] on ad hoc teams in a treasure hunt domain. A more theoretical ap-

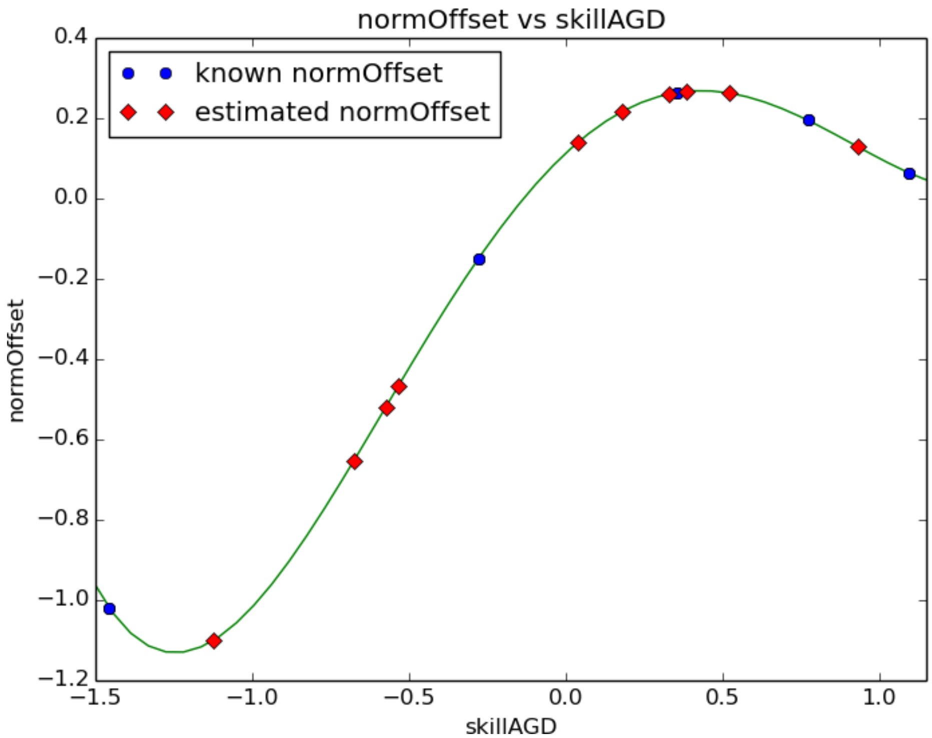

Fig. 4: Curve of `normOffset` vs `skillAGD` based on `normOffset` values normalizing `teamworkAGD` to 0 for Agent100, Agent80, Agent65, Agent50, and Agent30 from Table 8. Both data points used to generate the curve (blue dots) and points used to estimate `normOffset` (red diamonds) are shown.

proach is Wu et al.'s work [18] into ad hoc teams using stage games and biased adaptive play.

In the domain of robot soccer, Bowling and McCracken [3] measure the performance of a few ad hoc agents, where each ad hoc agent is given a playbook that differs from that of its teammates. In this domain, the teammates implicitly assign the ad hoc agent a role, and then react to it as they would any teammate. The ad hoc agent analyzes which plays work best over hundreds of games and predicts the roles that its teammates will play.

A popular way of ranking players based on relative skill is the Elo [4] rating system originally designed to rank chess players. While Elo only works in two player games, the TrueSkill [8] rating system allows for ranking players in games with multiple player teams. These ranking systems do not attempt to decouple a player's skill from its teamwork performance, and we are unaware of any such previously existing metrics that decouple skill and teamwork in an ad hoc teamwork setting.

An alternative and potentially promising way of estimating scores of drop-in player games is Liemhetcharat and Luo's adversarial synergy graph model [11] which has been used to estimate the scores of basketball games based on player lineups.

10 Conclusions

Drop-in player challenges serve as an exciting testbed for ad hoc teamwork, in which agents must adapt to a variety of new teammates without pre-coordination. These challenges provided an opportunity to evaluate agents' abilities to cooperate with new teammates to accomplish goals in complex tasks. They also served to encourage the participants in the challenges to reason about teamwork and what is actually necessary to coordinate a team.

This paper presents new metrics for assessing ad hoc teamwork performance, specifically attempting to isolate an agent's coordination and teamwork from its skill level, during drop-in player challenges. Additionally, the paper offers a prediction model for the scores of drop-in player games. This prediction model allows for smaller numbers of drop-in games being played when evaluating drop-in player challenge participants. When combined these contributions make it easier to study and perform research on ad hoc teamwork.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CNS-1330072, CNS-1305287, IIS-1637736, IIS-1651089), ONR (21C184-01), AFOSR (FA9550-14-1-0087), Raytheon, Toyota, AT&T, and Lockheed Martin. Peter Stone serves on the Board of Directors of, Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

1. Barrett, S., Stone, P.: An analysis framework for ad hoc teamwork tasks. In: AAMAS '12. (June 2012)
2. Barrett, S., Stone, P., Kraus, S.: Empirical evaluation of ad hoc teamwork in the pursuit domain. In: AAMAS '11. (May 2011)
3. Bowling, M., McCracken, P.: Coordination and adaptation in impromptu teams. In: AAAI. (2005)
4. Elo, A.: The rating of chess players, past and present (arco, new york). (1978)
5. Genter, K., Laue, T., Stone, P.: Benchmarking robot cooperation without pre-coordination in the robocup standard platform league drop-in player competition. In: Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-15). (September 2015)

6. Genter, K., Laue, T., Stone, P.: Three years of the robocup standard platform league drop-in player competition: Creating and maintaining a large scale ad hoc teamwork robotics competition. Autonomous Agents and Multi-Agent Systems (JAAMAS) **31**(4) (July 2017) 790–820
7. Grosz, B., Kraus, S.: Collaborative plans for complex group actions. Artificial Intelligence **86** (1996) 269–368
8. Herbrich, R., Minka, T., Graepel, T.: Trueskill$^{\text{TM}}$: a bayesian skill rating system. In: Proceedings of the 19th International Conference on Neural Information Processing Systems, MIT Press (2006) 569–576
9. Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K., Garvey, A.: The TAEMS White Paper (January 1999)
10. Jones, E., Browning, B., Dias, M.B., Argall, B., Veloso, M.M., Stentz, A.T.: Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In: ICRA. (May 2006) 570 – 575
11. Liemhetcharat, S., Luo, Y.: Applying the synergy graph model to human basketball. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. (2015) 1695–1696
12. Liemhetcharat, S., Veloso, M.: Modeling mutual capabilities in heterogeneous teams for role assignment. In: IROS '11. (2011) 3638 –3644
13. MacAlpine, P., Depinet, M., Liang, J., Stone, P.: UT Austin Villa: RoboCup 2014 3D simulation league competition and technical challenge champions. In: RoboCup-2014: Robot Soccer World Cup XVIII. Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin (2015)
14. MacAlpine, P., Genter, K., Barrett, S., Stone, P.: The RoboCup 2013 drop-in player challenges: Experiments in ad hoc teamwork. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (September 2014)
15. MacAlpine, P., Hanna, J., Liang, J., Stone, P.: UT Austin Villa: RoboCup 2015 3D simulation league competition and technical challenges champions. In: RoboCup-2015: Robot Soccer World Cup XIX. Lecture Notes in Artificial Intelligence. Springer Verlag, Berlin (2016)
16. Stone, P., Kaminka, G.A., Kraus, S., Rosenschein, J.S.: Ad hoc autonomous agent teams: Collaboration without pre-coordination. In: AAAI '10. (July 2010)
17. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research **7** (1997) 81–124
18. Wu, F., Zilberstein, S., Chen, X.: Online planning for ad hoc autonomous agent teams. In: IJCAI. (2011)