Adversarial Synergy Graph Model for Predicting Game Outcomes in Human Basketball

Somchaya Liemhetcharat
Institute for Infocomm Research
Agency for Science, Technology and Research
Singapore 138632, Singapore
liemhet-s@i2r.a-star.edu.sg

Yicheng Luo
National Junior College
Singapore 288913, Singapore
ethanluoyc@gmail.com

ABSTRACT

There are 30 teams in the National Basketball Association (NBA), and each team has a roster of 15 players. During a basketball match, a line-up of 5 players from one team plays against another line-up of 5 players from the opposing team. Typical approaches to predicting the game outcome use single-agent statistics, e.g., the percentage of successful 3-pointers of a player. In this paper, we adapt the Synergy Graph model – a multi-agent model that has been already applied to algorithms and robots – to human data from the NBA. The Synergy Graph model consists of a connected graph where vertices are the agents and edges represent the compatibility among agents, and single-agent capabilities are associated with each vertex. We learn an Adversarial Synergy Graph comprising NBA players using play-by-play data of past NBA games, with the goal of predicting which team will win when two line-ups play against each other. We show that the Adversarial Synergy Graph's prediction outperforms a regression model's prediction, thus illustrating the benefit of the Adversarial Synergy Graph model on human data.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

Team formation, performance modeling, learning, synergy, basketball, human teams

1. INTRODUCTION

A game of basketball involves a line-up of 5 players from one team versus another line-up of 5 players from another team. Each of these teams typically train and practice for months before the National Basketball Association (NBA) games, so the players have prior coordination within the team. However, it is difficult to anticipate and train against all other teams and line-ups in the NBA, even with some friendly games in the off-season. Thus, when considering two line-ups from opposing teams, it is generally true that they have not had prior coordination.

We are interested in modeling the interactions of these basketball players, in other to predict the outcome of the line-ups, i.e., which team will score more than the other. In general, we are interested to model *adversarial* multi-agent teams, in order to predict the overall performance.

Existing Synergy Graph models focus on fully collaborative multi-agent teams that coordinate on a shared task. In this paper, we contribute the Adversarial Synergy Graph model, where two opposing multi-agent teams are competing in a zero-sum game. Each agent is represented as a vertex in the Adversarial Synergy Graph, and the edges represent the compatibility among agents, within the same team, and across teams.

We use human basketball as our motivating domain, and demonstrate how the Adversarial Synergy Graph model is applied to basketball.We use three seasons of NBA play-by-play data (2007-08, 2008-09, and 2009-10) to train and test the Adversarial Synergy Graph model, and show that we outperform a linear regression approach. The only inputs required to train the Adversarial Synergy Graph are the player names of the two line-ups, and their point-difference; in contrast, detailed statistics on the line-ups are necessary for the training of linear regression. Thus, our approach is general and can be applied to many other domains without the need for domain-specific information.

Other approaches to predicting basketball performance, which we discuss later in the related work section, use detailed statistics, such as the percentage of successful field goals and the number of rebounds. Complex models are learned using these statistics, and frequently involve domain experts to extract relevant features. Further, these approaches focus on either single-player statistics, or teams as a whole, and do not model how the performance of a team varies based on its line-up. Our approach is different in that we model the performance of a particular line-up, in the presence of another line-up of the opposing team. By doing so, we predict different game outcomes depending on which line-up is chosen by the opposing team, even if the two line-ups have not intereacted with each other before (i.e., played a basketball game together).

Our approach is applicable to general multi-agent adversarial games, and we believe it is also of significant interest to basketball teams. Basketball team managers have to often choose to recruit players from other teams, and our model allows the managers to have an estimate of how the newly recruited player would perform in future games. Further, by incorporating the opponent's line-up in our model, basketball team managers would be able to select the optimal line-up that maximizes the probability of winning during a game.

The layout of our paper is as follows: Section 2 discusses related work in predicting basketball results, and previous Synergy Graph models. We give a background of the historical basketball data in Section 3. We present the Adversarial Synergy Graph model and learning algorithm in Section 4. We describe our experiments in Section 5 and conclude and discuss future work in Section 6.

2. RELATED WORK

Sports result predictions have been commonly studied in scientific research. Various approaches have been used to predict the outcomes of sports games. In some approaches, the predictions are made based on the experience and opinions of experts in basketball. In other approaches, data-mining techniques have been adopted to interpret the game results and for other assessments. Common data-mining techniques include artificial neural networks, decision trees, Bayesian methods, logistic regression, support vector machines, and fuzzy methods [3].

Buursma treats the prediction of football matches as a classification problem of "home win", "draw", "away win" and uses a few algorithms including decision forest, Bayesian networks, linear and logistic regression [1] using the Waikato Environment for Knowledge Analysis (WEKA) [4]. Kahn uses Artificial Neural Networks (ANN) to predict NFL football games [5].

As for basketball in particular, the outcome of the prediction can either be a direct classification of win/loss or as the regression of the point difference between the teams. To predict the point difference, one approach is to use multivariate linear regression. Miljkovic et. al predicted the outcome of the 2009-2010 season by using Bayesian inference to classify the win/loss of the games [11]. Furthermore, they used multivariate linear regression to predict the point difference (also known as spread) of the games. Cao used Simple Logistics Classifier, Artificial Neural Networks, SVM and Naive Bayes and found that the Simple Logistics Classifier produces the best results [2]. Trawinski proposed a fuzzy classification model to predict the ACB Basketball League results [12].

However, these approaches treat the team of players as a single agent. They do not consider the line-ups used during the game, e.g., a well-trained team of average players may outperform a team of star players that do not coordinate well.

The Synergy Graph model was introduced to capture the synergistic effects of agents in a collaborative multi-agent team [6, 10]. The Synergy Graph represents agents as vertices in a connected graph, and the edges represent the agents' task-based relationships, i.e., agents that are compatible and work better at a task have shorter pairwise distances. The Synergy Graph has been applied to software agents [10] and robots [7], as well as to configure a multi-robot team by selecting the robots' components [9, 8]. However, the Synergy Graph model has previously only been used to model the performance of a multi-agent team in a collaborative task. In this paper, we consider using the Synergy Graph model in an adversarial task (i.e., a zero-sum game) where two multi-agent teams are in competition.

3. BACKGROUND ON BASKETBALL DATA

The play-by-play data of the season from 2008 to 2009 of the National Basketball Association (NBA) is available online: we used `http://www.basketballgeek.com` as well as `http://www.basketball-reference.com`. The play-by-play data of game in the season is available as Comma Separated Values (CSV) files.

The play-by-play data describe the series of events that happen in a particular NBA game. These events include assists, blocks, steals, shots, free throws, rebounds and many other events that have statistical significance in evaluating the team's performance. Table 1 shows a sample of the play-by-play data. The first 5 columns (*h1-h5*) show the players of the line-up of the home team, and the next 5 columns (*o1-o5*) show the players of the line-up of the opposing team. The *time* column is the time remaining in a period (i.e., a quarter of the game). The *team* column is the team which the event on the *etype* column takes place.

To calculate the point difference between two particular line-ups, each CSV is parsed based on the substitution of a player, i.e., every segment of play-by-play data starts with the substitution (as represented by *etype sub*) of a player and ends with another substitution of a player. Table 1 shows one segment of the play-by-play data. We aggregate all the points scored by the team and the opponent team to calculate the point difference between the two teams. In the example shown in Table 1, the points scored by Cleveland Cavaliers (CLE) is 4 as there is 1 successful 2 point shot (*shot_2*) in row 2 and two successful free throws (*f_throw_1*) in row 8 and row 9, giving a total score of $2 + 2 = 4$ points. On the other hand, the points scored by Boston Celtics (BOS) is also 4 as there are two successful 2 point shots (*shot_2*) in row 3 and row 15 respectively. Hence the point difference between CLE and BOS is $4 - 4 = 0$ points. We repeat this process for all segments of the play-by-play data.

Besides the play-by-play data, we also retrieve the seasonal statistics, e.g., Field Goal Percentage (FGP), Free Throw Percentage (FTP) of the season, of the line-ups involved in the data from `http://stats.nba.com`. Each line-up has its own statistics, e.g., (House, Perkins, Garnett, Pierce, Allen) from Table 1 have a different FGP from (Wallace, West, James, Williams, Ilgauskas).

Many existing approaches consider the statistics of the entire team, e.g., the FGP of Cleveland Cavaliars (CLE), and use these team statistics to predict the game outcomes. Figure 1 shows the spread of some of the statistics of line-ups within a single team in the NBA; we illustrate the CLE team, but it is similar for other NBA teams. In the box-and-whisker plots, the boxes indicate the 25^{th} and 75^{th} percentiles, the red lines indicate the median, and the whiskers (i.e., the horizontal lines) indicate the maximum and minimum values. The chart of the left shows the Field Goal %, Free Throw %, and 3 Pointers %, and the chart on the right shows the number of rebounds, turnovers, fouls and assists. Since each line-up plays for a different amount of time through a game (and through a season), the values of rebounds, turnovers, fouls and assists have been proportionally scaled.

Hence, Figure 1 shows that there is a large variation of the various statistics among the line-ups within a team. As such, we believe that the line-ups play a big role in determining the outcome of a game. In particular, we believe that not only does the line-up of a team matter, the line-up of the opponent is important as well.

Our Adversarial Synergy Graph learning algorithm does not use the statistics of the two line-ups for training. These

	h1	h2	h3	h4	h5	o1	o2	o3	o4	o5	time	team	etype
1	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	06:38	BOS	sub
2	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	06:25	CLE	shot_2
3	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	06:11	BOS	shot_2
4	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:49	CLE	shot_miss
5	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:48	CLE	rebound
6	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:45	BOS	foul
7	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:45	BOS	timeout
8	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:45	CLE	f_throw_1
9	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:45	CLE	f_throw_1
10	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:32	CLE	foul
11	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:24	BOS	shot_miss
12	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:23	BOS	rebound
13	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:18	BOS	shot_miss
14	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	05:17	BOS	rebound
15	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	04:57	BOS	shot_2
16	House	Perkins	Garnett	Pierce	Allen	Wallace	West	James	Williams	Ilgauskas	04:57	CLE	foul
17	House	Perkins	Garnett	Pierce	Allen	West	James	Williams	Szczerbiak	Ilgauskas	04:57	CLE	sub

Table 1: One segment of the play-by-play data.

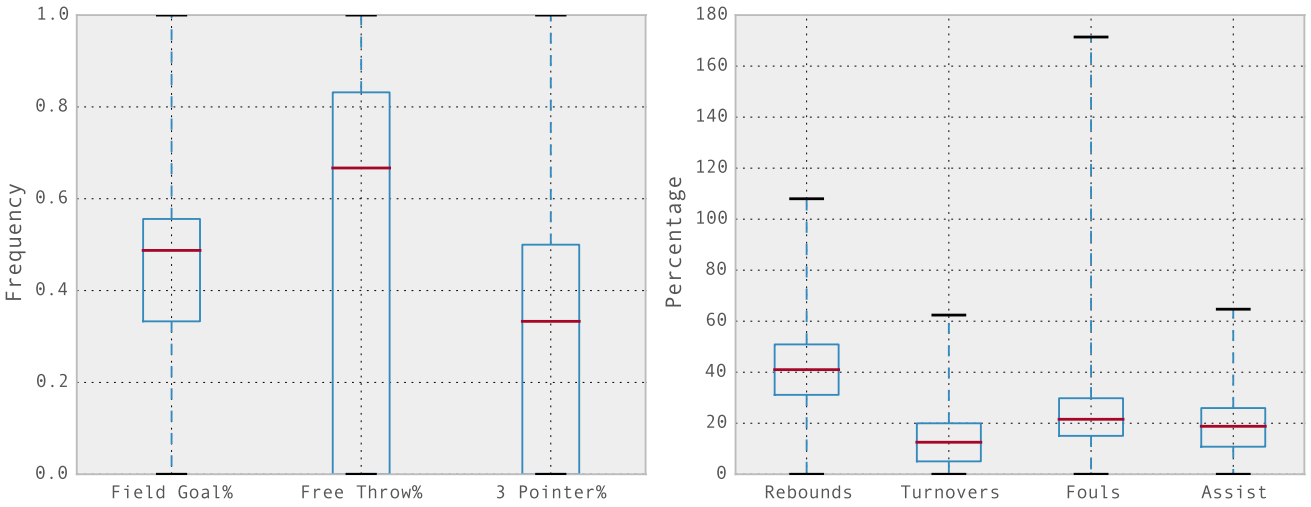

Figure 1: Box-and-whisker plots of various statistics of different line-ups within the Clevaland Cavaliars (CLE) team of the NBA.

statistics are used for the linear regression algorithm that we compare against. The data used for linear regression includes the two sets of the statistics of the two line-ups and the point difference produced by the two line-ups in the segment of data. We neglect the segments of play-by-play data that have line-ups whose statistics are not available at `http://stats.nba.com` due to little time played during the season.

4. OUR SYNERGY GRAPH APPROACH

The Synergy Graph model was introduced to model the performance of a multi-agent team performing collaborative tasks [6, 10]. In this paper, we adapt the Synergy Graph model to model the outcome of two adversarial multi-agent teams in a zero-sum game.

We first introduce the Adversarial Team Performance problem. We then formally define the Adversarial Synergy Graph model, and explain the key differences from the previously introduced Synergy Graph models. Next, we illustrate how the Adversarial Synergy Graph model is applied to human basketball, and describe how we learn an Adversarial Synergy Graph completely from data.

4.1 Adversarial Team Performance Problem

We begin with the definition of the set of agents and the definition of a team:

DEFINITION 4.1. *The **set of all agents** is* $\mathcal{A} = \{a_{1,1}, \ldots, a_{1,M_1}, a_{2,1} \ldots, a_{N,M_N}\}$*, where each* $a_{i,\alpha} \in \mathcal{A}$ *is an agent.*

The set of agents $\mathcal{A}$ contains all possible agents, and each agent $a_{i,\alpha}$ belongs to a group i. Teams are subset of agents that belong to the same group.

DEFINITION 4.2. *A **team** is any subset* $A_i \subseteq \mathcal{A}$ *such that* $\forall a_{j,\alpha} \in A_i, i = j$.

We are interested in the performance of two adversarial teams, i.e., the overall performance of a task given two com-

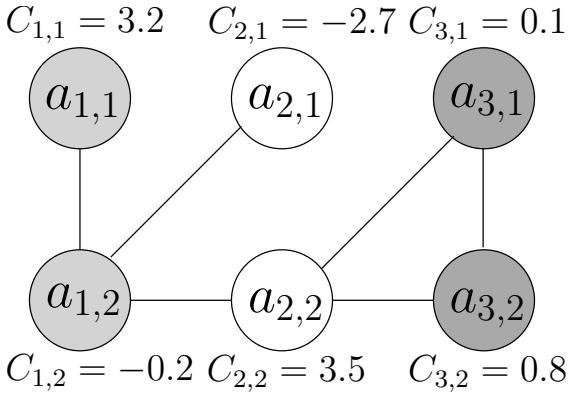

Figure 2: An example Adversarial Synergy Graph with 6 agents – 2 agents each in 3 groups.

petition teams. The performance may be positive or negative, indicating that the former or latter team performed better, similar to the notation of zero-sum games.

DEFINITION 4.3. *The **performance** of two adversarial teams A_i and A_j $(i \neq j)$ is $\mathcal{P}(A_i, A_j) \in \mathbb{R}$.*

Let D be the data set that contains some $\mathcal{P}(A_i, A_j)$, i.e., examples of performance of A_i and A_j.

The **goal** of the adversarial team performance problem is to model $\mathcal{P}_{A_i,A_j}$ for all A_i, A_j using D, in order to maximize:

$$\sum_{\mathcal{P}(A_i,A_j)\in D} \begin{cases} 1 & \text{if } \text{sign}(\hat{\mathcal{P}}(A_i, A_j)) = \text{sign}(\mathcal{P}(A_i, A_j)) \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{\mathcal{P}}(A_i, A_j)$ is the model prediction of $\mathcal{P}(A_i, A_j)$ and

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x \geq \epsilon \\ -1 & \text{if } x \leq -\epsilon \\ 0 & \text{otherwise} \end{cases}$$

Note that our use of $\text{sign}(x)$ is slightly different from the integer $\text{sign}(i)$, where $\text{sign}(i) = 0$ if and only if $i = 0$. In our case, since $x \in \mathbb{R}$ (performance is a real number), we use the threshold $(-\epsilon, \epsilon)$ to denote an equal performance between both adversarial teams, where $\epsilon \in \mathbb{R}^+$.

4.2 Adversarial Synergy Graph Model

We formally define the Adversarial Synergy Graph model:

DEFINITION 4.4. *An **Adversarial Synergy Graph** is a tuple (G, C), where:*

- *$G = (V, E)$ is a connected unweighted graph,*
- *$V = \mathcal{A}$, i.e., each agent is represented by a vertex,*
- *E are unweighted edges in G, and*
- *$C = \{C_{1,1}, \ldots, C_{1,M_1}, C_{2,1} \ldots, C_{N,M_N}\}$, where $C_{i,\alpha} \in \mathbb{R}$ is agent $a_{i,\alpha}$'s capability.*

The edges in the Adversarial Synergy Graph represent the task-based relationships among the agents, and we use the pairwise distances in order to compute the synergy of teams. A comprehensive description of task-based relationships and how they are modeled is presented in [10].

Figure 2 shows an example Adversarial Synergy Graph with 6 agents. The agents belong to three groups, where each group has two agents. Each agent has a unique capability, and the unweighted edges in the graph connect the agents. The shortest distance between agents is measured by the number of edges between them, e.g., the distance $d(a_{1,1}, a_{1,2}) = 1$ and $d(a_{1,1}, a_{3,2}) = 3$.

Using the Adversarial Synergy Graph structure and agent capabilities, we define the pairwise synergy among agents in the same team.

DEFINITION 4.5. *The **pairwise synergy** of two agents $a_{i,\alpha}$ and $a_{i,\beta}$ in the **same team** i is:*

$$\mathbb{S}_{2,team}(a_{i,\alpha}, a_{i,\beta}) = \phi(d(a_{i,\alpha}, a_{i,\beta})) \cdot (C_{i,\alpha} + C_{i,\beta})$$

where $d(a_{i,\alpha}, a_{i,\beta})$ is the shortest distance between the vertices representing the agents in the unweighted graph G (i.e., edges have a distance of 1), and $\phi : \mathbb{R} \to \mathbb{R}$ is the compatibility function.

The compatibility function ϕ maps larger distances to lower compatibility. Examples include $\phi_{\text{fraction}}(d) = \frac{1}{d}$ and $\phi_{\text{decay}}(d) = \exp\left(-\frac{d \ln 2}{h}\right)$. [10] provides details on the compatibility function, its motivation, as well as examples. In this paper, we use $\phi_{\text{fraction}}(d) = \frac{1}{d}$ as our compatibility function.

From Def. 4.5, the synergy of teammate agents increases when their individual capabilities ($C_{i,\alpha}$, $C_{i,\beta}$) increase, or when their pairwise distance ($d(a_{i,\alpha}, a_{i,\beta})$) decreases.

We now define the pairwise synergy among agents in different teams.

DEFINITION 4.6. *The **pairwise synergy** of two agents $a_{i,\alpha}$ and $a_{j,\beta}$ in **adversarial teams** $i \neq j$ is:*

$$\mathbb{S}_{2,adver}(a_{i,\alpha}, a_{j,\beta}) = \phi(d(a_{i,\alpha}, a_{j,\beta})) \cdot (C_{i,\alpha} - C_{j,\beta})$$

where $d(a_{i,\alpha}, a_{j,\beta})$ is the shortest distance between the vertices representing the agents in the unweighted graph G (i.e., edges have a distance of 1), and $\phi : \mathbb{R} \to \mathbb{R}$ is the compatibility function.

The main difference between Definitions 4.5 and 4.6 is in the usage of the agent capabilities. In Definition 4.5, the agent capabilities are summed because the agents are in the same team, so $\mathbb{S}_{2,\text{team}}(a_{i,\alpha}, a_{i,\beta}) = \mathbb{S}_{2,\text{team}}(a_{i,\beta}, a_{i,\alpha})$. In contrast, the difference in the agent capabilities is used in Definition 4.6, so $\mathbb{S}_{2,\text{adver}}(a_{i,\alpha}, a_{j,\beta}) = -\mathbb{S}_{2,\text{adver}}(a_{j,\beta}, a_{i,\alpha})$. Since the agents are in opposing teams, the difference in their capabilities plays a role in determining the overall team performance (i.e., which team will win).

From the pairwise synergy definitions, we now define the overall synergy function that computes the expected performance of two adversarial teams.

DEFINITION 4.7. *The **synergy** of two adversarial teams A_i and A_j $(i \neq j)$ is:*

$$\mathbb{S}(A_i, A_j) = \frac{1}{\binom{|A_i|+|A_j|}{2}} \left(\sum_{\{a_{i,\alpha}, a_{i,\beta}\} \in A_i} \mathbb{S}_{2,team}(a_{i,\alpha}, a_{i,\beta}) - \sum_{\{a_{j,\alpha}, a_{j,\beta}\} \in A_j} \mathbb{S}_{2,team}(a_{j,\alpha}, a_{j,\beta}) + \sum_{a_{i,\alpha} \in A_i, a_{j,\beta} \in A_j} \mathbb{S}_{2,adver}(a_{i,\alpha}, a_{j,\beta}) \right)$$

The synergy of the teams A_i, A_j is computed first by the difference of pairwise synergy of agents within the team A_i and agents within the team A_j, and then by the pairwise synergy of agents across both teams. Hence, both the capabilities and pairwise distances among the agents play a large role in affecting the final synergy result.

Also, note that $\mathbb{S}(A_i, A_j) = -\mathbb{S}(A_i, A_j)$, so $\mathbb{S}(A_i, A_j) > \epsilon$ indicates that A_i "won" A_j in the zero-sum game between A_i and A_j.

4.3 Differences from the Synergy Graph Model

There are a number of differences between the Adversarial Synergy Graph model we contribute in this paper, and the other Synergy Graph models introduced in the past. We highlight the main differences below:

1. The Adversarial Synergy Graph models interactions of agents in two adversarial teams;
2. The graph in the Adversarial Synergy Graph uses unweighted edges;
3. The agent capabilities are real numbers.

As described in Section 2, the previous Synergy Graph models focused on agents that collaborated in the same team for a common goal. In this paper, we consider *adversarial* teams, where two teams are in direct competition in a zero-sum game. It is interesting to consider "synergy" among agents in adversarial teams, since the term synergy is typically used in collaborative teams. We believe synergy is still an apt word for the adversarial case, because there is synergy within agents of each team, and also interactions between agents of opposing teams.

Secondly, our Adversarial Synergy Graph model uses unweighted edges. The initial Synergy Graph model used unweighted edges [6], but weighted edges were subsequently used to increase the space of representable domains (some examples of cases where unweighted edges cannot represent the task-based relationships are shown in [10]. However, we use unweighted edges in the Adversarial Synergy Graph model presented here for two main reasons: (1) it is the first model of adversarial interactions using Synergy Graphs, and (2) there are typically a large number of vertices when considering opponents, so the search space when learning the Adversarial Synergy Graph would be exponentially larger with weighted edges. We elaborate on (2) in the next subsection when we explain how we apply it to human basketball.

Thirdly, the Adversarial Synergy Graph model uses real numbers. The previous Synergy Graph models use Normally-distributed variables to capture the variability in agent and team performance. We believe that using Normal variables provides a more comprehensive model of team performance, but similar to reason (2) listed above, the large number of vertices in the Adversarial Synergy Graph increases the complexity of learning. We plan to consider weighted edges and Normally-distributed capabilities as future work.

4.4 Modeling Human Basketball Performance

To apply the Adversarial Synergy Graph model to human basketball, we consider the National Basketball Association (NBA) games. We chose the NBA for the following reasons:

- Statistics of players are readily available;
- Play-by-play data of NBA games are available;
- A basketball game involves 5 players versus 5 players, while each team has a roster of 15 players;
- The rules of basketball allows many player substitutions throughout one game.

We described the format of data available on NBA players and games in the previous section, and focus on how we apply the Adversarial Synergy Graph model to predict NBA results in this subsection.

Each agent $a_{i,\alpha}$ represents a player in the NBA (e.g., LeBron James). Each player is also associated with an NBA team (e.g., Cleveland Cavaliers). Hence, we create the Adversarial Synergy Graph using the players of the NBA and their associated teams.

The edges in the Adversarial Synergy Graph model represent the agents' task-based relationships, i.e., how well they play basketball with their teammates, and against their opponents. We will learn the Adversarial Synergy Graph structure from training data (explained in the next subsection).

The agent capabilities $C_{i,\alpha}$ represent the contribution of the player to the overall team performance. Various statistics of basketball players are available, such as their field goal percentage (FGP), i.e., the probability that a field goal thrown by the player enters the hoop. However, these statistics are typically offense-oriented and do not capture the contributions of defensive players well. Hence, we will learn the agent capabilities from data as well.

The training data provided to learn the Adversarial Synergy Graph model is the following:

$$((h_1, h_2, h_3, h_4, h_5), (o_1, o_2, o_3, o_4, o_5), ptd)$$

where $h_1, \ldots, h_5$ are the names of the players on the line-up of home team, and $o_1, \ldots, o_5$ are the players on the line-up of the opposing team. Each line-up in basketball consists of exactly 5 players. ptd is the point-difference between these two line-ups.

In a typical basketball game, there are many line-ups chosen by each team, so a single game yields many unique $((h_1, \ldots, h_5), (o_1, \ldots, l_5), ptd)$ tuples. We chose to use the point-difference as the team performance because it is zero-sum (i.e., a positive value indicates that the home team scored more points than the opposing team, and vice versa). Furthermore, point-difference is time-independent, in that a line-up pair that played for 10 minutes can be compared equivalently to a line-up pair that played for 3 minutes, since both teams had equal opportunities to score and defend. In contrast, measuring points scored or goals attempted would be non zero-sum and time-dependent.

It is very interesting that we only use the line-up names and point difference as input to our Adversarial Synergy Graph learning algorithm – we do not need other information such as individual player statistics, or other basketball-specific information. We believe that this is one of the main unique attributes of the Synergy Graph approach – very little domain-specific knowledge is required, only the two line-ups and the final outcome.

4.5 Learning the Adversarial Synergy Graph from Data

To learn the Adversarial Synergy Graph, we use a similar technique that has been previously employed on previous Synergy Graph models (e.g., [10]):

1. We initialize a random connected graph structure with the number of vertices matching the number of agents;
2. From the graph structure, we learn the agent capabilities with data;
3. We iteratively improve the graph structure and learn agent capabilities.

Algorithm 1 shows our learning algorithm:

Algorithm 1 Learn an Adversarial Synergy Graph for the agent set $\mathcal{A}$, using the training set D_{train}

```
LearnAdSynGraph(A, D_train)
 1: // D_train = {((h_1, ..., h_5), (o_1, ..., o_5), ptd), ...}
 2: // Create a random connected unweighted graph
 3: G ← RandomGraph(|A|)
 4: // Learn the agent capabilities using G and D_train
 5: C ← LearnCapabilities(G, D_train)
 6: // Form the initial Adversarial Synergy Graph
 7: S ← (G, C)
 8: // Compute the initial training error
 9: e ← ComputeError(S, D_train)
10: // Simulated annealing
11: k ← 0
12: while k < K do
13:    G' ← RandomNeighborStructure(G)
14:    C' ← LearnCapabilities(G', D_train)
15:    S' ← (G', C')
16:    e' ← ComputeError(S', D_train)
17:    if Accept(e, e') then
18:       S ← S'
19:       e ← e'
20:       k ← 0
21:    else
22:       k ← k + 1
23:    end if
24: end while
25: return S
```

4.5.1 Creating the initial Adversarial Synergy Graph

The first step of the algorithm involves creating an initial "guess" of the Adversarial Synergy Graph. To do so, Algorithm 1 first generates a random graph structure (line 2), based on the set of agents $\mathcal{A}$. Specifically, it creates $|\mathcal{A}|$ vertices, and randomly generates edges between all pairs of vertices. Each possible edge has a probability p of being generated, and the graph structure generated is tested to ensure that the graph is connected, i.e., there exists a path from any vertex to any other vertex.

After the initial graph G is created, the agent capabilities C is learned (line 5 of Algorithm 1). We will describe the LearnCapabilities function in the next subsection.

The training error of the initial Adversarial Synergy Graph is then computed (line 9), which uses the root-mean squared error of the predicted point difference:

$$\text{ComputeError}(S, D) = \sqrt{\frac{\sum_{(A_i, A_j, ptd) \in D} (\mathbb{S}(A_i, A_j) - ptd)^2}{|D|}}$$

4.5.2 Learning the agents' capabilities

From the unweighted graph structure G and the training data D_{train}, we learn the agent capabilities C. To do so, we solve a series of linear equations using Definitions 4.5, 4.6, and 4.7.

Notice that the Synergy equations are all linear, and we can expand them as follows ($k = \frac{1}{\binom{|A_i|+|A_j|}{2}}$):

$$\begin{aligned}
\mathbb{S}(A_i, A_j) = k \cdot & \sum_{\{a_{i,\alpha}, a_{i,\beta}\} \in A_i} \phi(d(a_{i,\alpha}, a_{i,\beta}))(C_{i,\alpha} + C_{i,\beta}) \\
- k \cdot & \sum_{\{a_{j,\alpha}, a_{j,\beta}\} \in A_j} \phi(d(a_{j,\alpha}, a_{j,\beta}))(C_{j,\alpha} + C_{j,\beta}) \\
+ k \cdot & \sum_{a_{i,\alpha} \in A_i, a_{j,\beta} \in A_j} \phi(d(a_{i,\alpha}, a_{j,\beta}))(C_{i,\alpha} - C_{j,\beta}) \\
\mathbb{S}(A_i, A_j) = & \sum_{a_{i,\alpha} \in A_i} k_{i,\alpha} \cdot C_{i,\alpha} + \sum_{a_{j,\beta} \in A_j} k_{j,\beta} \cdot C_{j,\beta}
\end{aligned}$$

In particular, from the training data D_{train}, we know what value $\mathbb{S}(A_i, A_j)$ should be, and so each tuple in D_{train} forms a linear equation where the only unknowns are the agent capabilities $C_{i,\alpha}$ and $C_{j,\beta}$, since all $k_{i,\alpha}$ and $k_{j,\beta}$ can be computed from the graph structure (it only depends on k and the distances between vertices in the graph). Hence, we can find the least-squares solution to these series of linear equations, and learn the agents' capabilities.

4.5.3 Iteratively improving the Adversarial Synergy Graph

To learn the final Adversarial Synergy Graph, we iteratively improve the initial guess using simulated annealing. We randomly create neighbor graph structures, i.e., connected unweighted graphs that differ from the original by either adding or removing an edge.

Using the neighbor graph structure, we learn the agents' capabilities (described in the previous subsection), and combine them to form a neighbor Adversarial Synergy Graph (note that the Adversarial Synergy Graph consists of only the graph structure and the agents' capabilities). We then compute the error in the predicted point difference of the neighbor Adversarial Synergy Graph. The error of the current and neighbor Adversarial Synergy Graphs are compared, and the neighbor is accepted subject to the temperature schedule of the simulated annealing.

The key difference between the simulated annealing algorithm in this paper versus previous Synergy Graph work (e.g., [10]) is that we do not use a fixed number of iterations of simulated annealing. We terminate the simulated annealing only when no neighbors have been accepted for K iterations. We chose to modify the simulated annealing algorithm in this way because the space of Adversarial Synergy Graph structures is very large ($\mathcal{O}(2^{|\mathcal{A}|})$), and $|\mathcal{A}|$ was also much larger than in previous Synergy Graph work (which we elaborate in the next section).

5. EXPERIMENTS AND RESULTS

In this section, we describe the experiments we conducted to evaluate the Adversarial Synergy Graph model in predicting the outcomes of human basketball games. We first discuss how we parsed the historical play-by-play data to provide input to the Synergy Graph learning algorithm. We next describe the linear regression algorithm that we used as a benchmark. Then, we present our experimental results and analysis.

5.1 Parsing Historical Data

We used the NBA play-by-play data of the 2007-2008, 2008-2009, and 2009-2010 seasons, which comprises 1183, 1176, 1215 games respectively. Within each game, the two teams frequently changed their line-ups, and we extracted the $((h_1, h_2, h_3, h_4, h_5), (o_1, o_2, o_3, o_4, o_5), ptd)$ tuples, where h_i and o_j are the names of the players in the home and opposing teams respectively. In the 2008-2009 season for example, there were 444 unique players (some players did not play any games at all and were excluded), and 60235 tuples.

Since creating an Adversarial Synergy Graph of all the NBA players would require a graph of 450 vertices per season, we decided to use a subset of the data for our experiments, of ∼150 players and ∼1000 line-up tuples (the exact numbers varied a little depending on the season). We chose such a number so that the Adversarial Synergy Graph would still be large (150 vertices) but not prohibitively so. The previous Synergy Graph work typically had around 15-25 vertices, so this was six-fold increase in the vertex size. The 1000 line-up tuples were selected such that their time played was in the top 25%; line-ups that played for longer durations had more consistent results since the players had more time on the court to attempt to score and defend against shots.

Each season was tested independently, so for the 1000 tuples of every season, we performed 10-fold cross validation, i.e., in each fold, 90% of the data was used for training and 10% for testing. The Synergy Graph learning algorithm used the training sets to learn an Adversarial Synergy Graph, and we present our results on the test sets. The Adversarial Synergy Graph predicts the expected point-difference between the line-ups, and we thresholded the results into win/draw/loss, using $\epsilon = 0.5$, e.g., a team wins if the predicted point difference $\geq \epsilon$, and loses if the prediction $\leq -\epsilon$.

5.2 Comparing to Linear Regression

To benchmark our Adversarial Synergy Graph, we compared against linear regression, because it is commonly used in predicting sports results. As mentioned above, we retrieved the historical statistics of the line-ups from the website `http://stats.nba.com` and used these statistics for linear regression. In particular, we used these features for the input to linear regression:

- Personal Fouls (PF)
- Personal Fouls Drawn (PFD)
- Total Rebounds (REB)
- Defensive Rebounds (DREB)
- Offensive Rebounds (OREB)
- Steals (STL)
- Turnovers (TOV)
- Assists (AST)
- Blocks (BLK)
- Blocks Against (BLKA)
- 3-Pointers Attempted (FG3A)
- 3-Pointers Made (FG3M)

Approach	NBA Season		
	2007-2008	2008-2009	2009-2010
Ad. Syn.	$\mathbf{63.2 \pm 3.8}\%$	$\mathbf{63.2 \pm 4.4}\%$	$\mathbf{69.9 \pm 5.6}\%$
L. Regr.	$46.2 \pm 2.3\%$	$40.4 \pm 7.0\%$	$38.0 \pm 4.8\%$
Random	$42.1 \pm 4.3\%$	$45.2 \pm 6.5\%$	$44.2 \pm 4.2\%$
All Win	$44.1 \pm 2.8\%$	$45.8 \pm 5.2\%$	$46.1 \pm 4.8\%$
All Lose	$45.1 \pm 3.3\%$	$45.6 \pm 6.3\%$	$46.9 \pm 5.4\%$
All Draw	$10.8 \pm 3.5\%$	$8.6 \pm 3.4\%$	$7.1 \pm 2.2\%$

Table 2: Test Accuracy of Adversarial Synergy Graph (Ad. Syn), linear regression (L. Regr.) and other approaches in predicting win, loss, and draw in line-ups of basketball.

- 3-Pointers Percentage (FG3P)
- Field Goals Attempted (FGA)
- Field Goals Made (FGM)
- Field Goals Percentage (FGP)
- Free Throws Attempted (FTA)
- Free Throws Made (FTP)
- Free Throws Percentage (FTP)

These statistics were chosen because they are the most common statistics available for players and line-ups. Note that we retrieved the statistics for the line-ups and not for the overall team, as the variance in these statistics are rather large within the team. To our knowledge, most existing work focus on the statistics of the entire team, so considering the statistics of the line-ups for linear regression is already a novel step, and should aid in the final prediction of the outcome of the entire game. However, in the context of the experiments for this paper, since we were predicting the outcome when two line-ups face off against each other, we believe that it makes more sense to consider the statistics of the line-ups themselves, rather than the overall statistics of the teams.

The linear regression model was trained using the statistics listed above for both line-ups, and the point-difference. Some line-ups did not play for a significant amount of time during the entire season, and their statistics were not available online, so we excluded those line-ups from the training and testing data sets. During testing, the statistics were again used to generate a prediction of the expected point-difference, and then thresholded into win/draw/loss (identical to our thresholding for the Adversarial Synergy Graph prediction). Note that the linear regression technique requires these statistics of the line-ups. Our Adversarial Synergy Graph approach does not. We only require the identities of the players in the line-ups, and the point difference between the two line-ups.

5.3 Experimental Results

Table 2 shows the prediction accuracy in the test set for the 10-fold cross validation, of the Adversarial Synergy Graph (Ad. Syn.) and linear regression (L. Regr.), as well as benchmarks that always return win, lose or draw regardless of the line-ups, and a policy that randomly returned win, lose or draw. The random policy picked the outcome based on the proportions of win/lose/draw in the training set.

Our results clearly show that our Adversarial Synergy Graph approach outperforms linear regression and the other benchmarks. For example, our Adversarial Synergy Graph had an average accuracy of $65.5 \pm 5.5\%$ compared with linear regression's $41.5 \pm 6.0\%$. The random and static policies had similar performance to linear regression, which shows that considering the line-ups' statistics may not significantly improve the prediction of outcomes in 5v5 line-ups.

Thus, the Adversarial Synergy Graph returns a result that is much better than random, and yet only requires limited input for training, i.e., the members of both line-ups and their point difference. During testing, our approach only requires the identities of the line-ups. We do not require any other external information such as the statistics of the players of the line-ups, which other techniques such as linear regression require.

6. CONCLUSION AND FUTURE WORK

We introduced the Adversarial Synergy Graph (AdSyn) model to predict the outcome of a adversarial zero-sum game, that is an improvement over previous Synergy Graph models, in that opponents are also represented. We applied the Adversarial Synergy Graph model to human basketball data from the National Basketball Association (NBA) games, where each human player is a vertex in the Adversarial Synergy Graph.

The Adversarial Synergy Graph was learned using the 2007-08, 2008-09, 2009-10 NBA data, and we outperformed the benchmark of linear regression. Furthermore, the AdSyn requires *only* the training data of the line-up members of both teams and the point difference. In contrast, linear regression and other similar approaches in the literature require a much larger set of highly domain-specific information.

It is interesting that the Adversarial Synergy Graph model can predict the outcome of line-ups that have not previously met (or coordinated) in games, by modeling the interactions of the players in previous games in the training data set. Thus, it demonstrates that the Adversarial Synergy Graph model is suitable for modeling complex interactions among agents without prior coordination in adversarial tasks. In comparison, methods that rely heavily on historical data would be unable to make any predictions on new combinations of lineups. In particular, most, if not all, of the existing literature on sports predictions only consider predicting the outcomes of whole games, and typically consider the statistics of the team and not the individual line-ups.

As future work, we aim to learn a Adversarial Synergy Graph of all 444 players, and improve the prediction accuracy of our approach.

7. REFERENCES

[1] D. Buursma. Predicting Sports Events from Past Results. In *14th Twente Student Conference on IT*, 2010.

[2] C. Cao. Sports data mining technology used in basketball outcome prediction. Master's thesis, Dublin Institute of Technology, 2012.

[3] M. Haghighat, H. Rastegari, and N. Nourafza. A Review of Data Mining Techniques for Result Prediction in Sports. *Advances in Computer Science: an International Journal*, 2(5):7–12, 2013.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), 2009.

[5] J. Kahn. Neural Network Prediction of NFL Football Games. University of Wisconsin – Electrical and Computer Engineering Department, 2003.

[6] S. Liemhetcharat and M. Veloso. Modeling and Learning Synergy for Team Formation with Heterogeneous Agents. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 365–375, 2012.

[7] S. Liemhetcharat and M. Veloso. Weighted Synergy Graphs for Role Assignment in Ad Hoc Heterogeneous Robot Teams. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5247–5254, 2012.

[8] S. Liemhetcharat and M. Veloso. Forming an Effective Multi-Robot Team Robust to Failures. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5240–5245, 2013.

[9] S. Liemhetcharat and M. Veloso. Synergy Graphs for Configuring Robot Team Members. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 111–118, 2013.

[10] S. Liemhetcharat and M. Veloso. Weighted Synergy Graphs for Effective Team Formation with Heterogeneous Ad Hoc Agents. *Journal of Artificial Intelligence*, 208(2014):41–65, 2014.

[11] D. Miljkovic, L. Gajic, A. Kovacevic, and Z. Konjovic. The Use of Data Mining for Basketball Matches Outcomes Prediction. In *International Symposium on Intelligent Systems and Informatics*, pages 309–312, 2010.

[12] K. Trawinski. A Fuzzy Classification System for Prediction of the Results of the Basketball Games. In *IEEE International Conference on Fuzzy Systems*, pages 1–7, 2010.