Expanding the TrueSkill Algorithm Using In-Game Events

by

Justin D'Astous

Supervisor

Ryan Browne
Associate Professor,
Dept. of Statistics and Actuarial Science,
University of Waterloo

Research project submitted in fulfillment
of the requirements for the degree of
Master of Mathematics (MMath) in
Computational Mathematics
University of Waterloo

Waterloo, Ontario, Canada, 2022

Abstract

The Elo rating system was created in 1978 to better model the outcome of chess matches and estimate player skill. Player are rated based on the outcome of their previous games and the perceived skill of their previous opponents. The model uses the difference in estimated skill between players to calculate the probability of future game outcomes. The idea was extended to multiplayer matches in the TrueSkill algorithm, originally used to rank players in multiplayer Halo matches. This algorithm predicts the outcome of games based on the sum of individual player's skill ratings. This assumes players play most if not all of the games and are always playing the same opponents during the game. This paper explores implementing the TrueSkill algorithm using the outcome of individual non-scoring events within a game to rank players instead of just the outcome of the game. Each in-game event is weighted based on its estimated impact on the outcome of the game. This approach could allow us to rank players in fluid games where players only play a fraction of the game. This approach also helps differentiate between teammates which always play on the same team. This paper uses publicly available data for hockey games in the NHL to test the model.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Elo rating system was created in 1978 to better model the outcome of chess matches and estimate player skill. Player are rated based on the outcome of their previous games and the perceived skill of their previous opponents. The model uses the difference in estimated skill between players to calculate the probability of future game outcomes. Glicko models improved this simple player rating system by modeling player skill using Gaussian random variables. Each player is assigned a skill mean and variance. The idea was extended to multiplayer matches in the TrueSkill algorithm, originally used to rank players in multiplayer Halo matches. This algorithm predicts the outcome of games based on the sum of individual player's skill ratings. This assumes players play most if not all of the games and are always playing the same opponents during the game.

This paper explores implementing the TrueSkill algorithm using the outcome of individual events within a game to rank players instead of just the outcome of the game. Events that directly affect the outcome of the game are considered as well as events that indirectly affect the outcome. Each in-game event is weighted based on its estimated impact on the outcome of the game. This approach could allow us to rank players in fluid games where players only play a fraction of the game. This approach also helps differentiate between teammates which always play on the same team. Using in-game events also increases the number of outcomes passed into the TrueSkill model.

This paper uses publicly available data for hockey games in the NHL to test the model. All in-game events are weighted using multiple pre-existing hockey models that quantify the impact of in game events as well as the quality of shots. The model was run in R and the code is available upon request.

Chapter 2

Background

2.1 Elo Ratings

The Elo rating system was proposed as an improved way to rank chess players by the Frenchman Arpad Elo in 1978 [3]. Using the ratings, one can model the probability of a player winning any given matchup. In the system, every new player is given the same rating of 1000. The rating then gets updated after each head-to-head match based on the outcome of the match and the rating discrepancy between the two opponents.

Contributions by Mark Glickman [5] expanded the Elo rating system by modeling the player skill rating as a Gaussian random variable. These new models are referred to as the Glicko models. Adding in uncertainty to a player's rating greatly improved how the model handled newer players. New player ratings are based on few games so the outcomes of their games are less predictable. In the Elo system, players with less than 20 games were considered to have a provisional rating. In the Glicko model, the variance of a new player's rating properly represents the uncertainty of contests with new players. The game outcome can greatly effect a new player's rating while an established player's rating is less effected.

2.2 TrueSkill

Microsoft proposed a new algorithm named TrueSkill [6] that extends the work of Elo and Glickman to team based games. In particular, TrueSkill was first used to rate players in

their online multiplayer shooting game Halo 2 (Herbrich, Minka & Graepel, 2007). The player's skill rating would then be used by the Halo 2 matchmaking system. Players often queued up for a multiplayer game on their own. Microsoft's matchmaking system would then group players into teams that would create balanced and interesting matches. It was later offered to all games on Microsoft's Xbox 360 Live platform.

Similarly to the Glicko models, TrueSkill models player's skill rating as a Gaussian random variable. It then models each player's performance as the player's skill plus a Gaussian noise variable representing uncertainty within each game. The team performance is set as the sum of the performances of the members of the team. The difference in team performances are used to calculate the likeliness of the game outcome. The only possible outcomes are a win, a draw or a loss. Player's skill mean and variance are then updated accordingly (see section 2.2.1). TrueSkill also works in cases where more than 2 teams are competing by ranking the teams. Then teams are compared pair-wise, where the higher placed team is considered the winner between the two unless they are tied.

Microsoft has since released an updated TrueSkill 2 [14] which tries to account for more factors that could effect the skill of a player or a team. For example, the variance of a player's skill now increases over time if they are inactive. The performance of a player in a team can also be adjusted if their are playing as part of a squad, if they are more experience or based on their previous individual in-game statistics.

2.2.1 Simple Chess Example

Let's take a look at how TrueSkill updates player ratings by considering the simple case of a two person chess match. We will further simplify this example by assuming no ties. The pre-match skill S_i of players 1 & 2 in our example are assumed to be $S_i \sim N(\mu_i, \sigma_i^2)$ where $\mu_1 = 1400, \sigma_1 = 40, \mu_2 = 1200$ and $\sigma_2 = 150$. μ_i represents the current estimate for the skill of player i. A higher value means a higher skill and a higher likelihood of winning games. This value is adjusted after each game outcome to better reflect the player's estimated skill level. σ_i represents the uncertainty around the estimated skill level μ_i. It's value will decrease as more games are input into the model and more information on the player's skill is observed.

We can see in our example that player 1 has a higher mean skill level than player 2. Player 1 also is a more established with less variance in his rating.

In the TrueSkill model, each player's performance P_i is modeled to be normally distributed around the player's skill S_i plus a noise variable with constant variance β^2. The

added noise represents the variability of a players performance within a given game. It represents additional game outcome uncertainty. The value of β will determine how spread out player's ratings will be. A small β causes a small difference in skill to result in a large win probability, and *vis versa*. The performance variables are modeled as $P_i \sim N(\mu_i, \sigma_i^2 + \beta^2)$. In chess, they use $\beta = 200$ for their ratings [3], so we will do the same for our example.

The winner of a game is the player with the higher realised performance value p_i. Whereas the probability of a player winning is determined by the distribution of $P_1 - P_2$. For our example, we will consider the less likely outcome that player 2 won the match. To update the true skill, we first need to calculate the mean difference in team performance between the winner and the loser. The difference in performance is $\Delta = \mu_2 - \mu_1 = 1200 - 1400 = -200$.

We can then update both player's rating mean and variance. The TrueSkill model is a Bayesian model. Each observed outcome is approached with a prior describing the probability of the outcomes. In this case, the prior is the players original rating mean and variance. Once the outcome is observed, the likelihood function is calculated using the prior. The posterior, or updated player mean and variance, is calculated by combining the prior and the likelihood.

To do this, we first calculate intermediary variables c, v, w. The c variable represents the total standard deviation of the player performances within the game. The ratio $\frac{\Delta}{c}$, modeled as a standard $N(0, 1)$ Gaussian distribution, describes the likelihood of the outcome. That likelihood is then used to update the skill means and variance. The v variable is used scale to scale how much the skill mean will be updated. The more unlikely the outcome, the higher the value will be and the more the skill means will be shifted. The w variable is similar to v, but scales how much the skill variance will be reduced from the outcome. Similarly to v, the more unlikely the outcome, the higher the value. The more unlikely the outcome, the less the skill variance will be reduced.

$$
\begin{aligned}
c &= \sqrt{2\beta^2 + \sigma_1^2 + \sigma_2^2} = \sqrt{2 \times 200^2 + 40^2 + 150^2} = 285.92 \\
v &= \frac{\phi\left(\Delta/c\right)}{\Phi\left(\Delta/c\right)} = \frac{\phi\left(-200/285.92\right)}{\Phi\left(-200/285.92\right)} = 1.2901 \\
w &= v\left(v + \frac{\Delta}{c}\right) = 1.2901\left(1.2901 + \frac{-200}{285.92}\right) = 0.7619 \\
I_1 &= (-1)^{\mathbb{1}_{\text{Player 1 Lost}}} = -1 \\
I_2 &= (-1)^{\mathbb{1}_{\text{Player 2 Lost}}} = 1
\end{aligned}
$$

Now we can perform the updates. In the equations, the dynamics variance τ is a constant variance added to each player's skill to ensure the variance does not tend to 0. It can represent inherent variance in a player's skill over time. The value of τ can be increased if a player has not played in a long time and his rating estimation is deemed to be less accurate. In this example, we will set $\tau = 0$ to keep the calculations simpler.

$$\begin{aligned}\mu_{1_{new}} &= \mu_1 + I_1 \times v\frac{\sigma_1^2 + \tau^2}{c} \\ &= 1400 + (-1) \times 1.2901\frac{40^2 + 0^2}{285.92} \\ &= 1392.78\end{aligned} \qquad \begin{aligned}\mu_{2_{new}} &= \mu_2 + I_2 \times v\frac{\sigma_2^2 + \tau^2}{c} \\ &= 1200 + 1 \times 1.2901\frac{150^2 + 0^2}{285.92} \\ &= 1301.52\end{aligned}$$

$$\begin{aligned}\sigma_{1_{new}}^2 &= \left(\sigma_1^2 + \tau^2\right)\left(1 - w\frac{\sigma_1^2 + \tau^2}{c^2}\right) \\ &= \left(40^2 + 0^2\right)\left(1 - 0.7619\frac{40^2 + 0^2}{285.92^2}\right) \\ &= 1576.14\end{aligned} \qquad \begin{aligned}\sigma_{2_{new}}^2 &= \left(\sigma_2^2 + \tau^2\right)\left(1 - w\frac{\sigma_2^2 + \tau^2}{c^2}\right) \\ &= \left(150^2 + 0^2\right)\left(1 - 0.7619\frac{150^2 + 0^2}{285.92^2}\right) \\ &= 17781.83\end{aligned}$$

$$\sigma_{1_{new}} = 39.70 \qquad \sigma_{2_{new}} = 133.35$$

We see that as a result of the game, the skill of player 2 has greatly increased from 1200 to 1301.52 and their σ has decreased a bit from 150 to 133.35. On the other hand, player 1 saw a small change in skill and variance. In response to an unlikely outcome, the player with the larger variance is more heavily impacted as they are more likely to be further from their mean skill level. In response to a likely outcome, the TrueSkill model will make larger updates to the variance of the players since their mean skill was more likely closer to it's true value. If both players kept on playing and their ratings were updating based on the outcomes, their ratings would converge to their true underlying skill. An example though simulation is shown in section 4.1.1.

2.2.2 Head to Head Team Games

When using TrueSkill to measure player skill in head to head team games, the methodology is similar to the example in section 2.2.1. We first determine the team performance using

the individual player skill levels. Let $\mathcal{A} = \{a_i\}$ to be the set players in team A and $\mathcal{B} = \{b_i\}$ to be the set of players in team B. Each team's performance is the sum of each player's individual performance. This is represented by $P_{\mathcal{A}} \sim N\left(\sum_{a_i \in \mathcal{A}} \mu_{a_i}, \sum_{a_i \in \mathcal{A}} (\sigma_{a_i}^2 + \beta^2)\right)$ and $P_{\mathcal{B}} \sim N\left(\sum_{b_i \in \mathcal{B}} \mu_{b_i}, \sum_{b_i \in \mathcal{B}} (\sigma_{b_i}^2 + \beta^2)\right)$. The winner of the game is the team with the highest realized performance.

When updating player ratings, e.g. if team A won, you would need to make the following adjustments to the team performance Δ and to the value of the c variable.

$$\Delta = \sum_{a_i \in \mathcal{A}} \mu_{a_i} - \sum_{b_i \in \mathcal{B}} \mu_{b_i}$$

$$c = \sqrt{(|\mathcal{A}| + |\mathcal{B}|)\beta^2 + \sum_{a_i \in \mathcal{A}} \sigma_{a_i}^2 + \sum_{b_i \in \mathcal{B}} \sigma_{b_i}^2}$$

The rest of the calculations remain the same as shown in the example above. Each player's mean and variance are updated after each game.

2.2.3 Multi-Team Games

To update ratings in the case where more than 2 teams are involved in the game, then it is necessary to iterate pairwise over the teams. All teams will need to be ordered in the standings reflecting their performance in the game. When comparing two teams, the team with the higher standing at the end of the game will be considered the winner. Teams can be tied, in which case the rating updates are more complex. The Δ calculation also changes to allow for ties. Since this will be outside the scope of this report, we will not discuss it further. For more information, you can look at the TrueSkill paper [6].

2.2.4 TrueSkill Applied to Sports

TrueSkill has been used to rank players or teams in a sporting context. For tennis, Gustavo Landfried suggested to created multiple skill levels for a single player [11]. i.e. A general skill level representing the skill of a player, and the other skill level representing the type of court on which the match was played. As a tennis player accumulated more games on

multiple court types, his general skill might differ from the court type specific skill. For example, the player "Rafael Nadal" is known for being dominant on clay so the model would be able to account for court type. This means that, when calculating his probability of winning a game, the player with skill level equal to "Rafael Nadal on clay" would be stronger than just "Rafael Nadal" or "Rafael Nadal on grass". It is an elegant method to update a player's rating based on their opponent as well as the context in which the game was played.

Another application to sports attempted to account for home court advantage and the winning margin when ranking tennis players as well as football and hockey teams [8]. They also used Gibbs sampling and batch processing to marginally increase the predictive power of their rankings. When ranking teams, they considered them to be a single entity and did not attempt to assign rankings to individual players in each team.

2.3 Hockey Analytics

Many new metrics and models have been created to try to quantify what happens on the ice of an NHL game. Hockey is a complex game to model due to multiple reasons. The game is very fluid with possession of the puck often changing sides very quickly, making it hard to separate individual plays. Players do not spend extended amounts of the time on the ice. As a result, the players on the ice change very regularly, making it hard to track how much someone is responsible for the outcome of a game. To attempt to overcome these obstacles, many different approaches have been taken.

2.3.1 Player Value Models

Some models attempt to evaluate players by quantifying the number of goals or wins that a player is contributing to their team. Most of these models will attempt to quantify the number of goals or wins a player contributes over a replacement level player, *i.e.* wins over replacement or goals over replacement. A replacement level player is defined as a player that is on the fringe of most teams and could be easily acquired or replaced. An example of such a model would be the one created by Luke Solberg, also known as EvolvingWild [23]. The model uses a mixture of a large number of on-ice metrics to evaluate players. Linear regression and machine learning is used to determine which metrics offered the most insight. The goal of this model was to predict a player's future value, therefore, a player's future skill is projected using age curves [22] which describe a player's perceived

skill over their career. Other models have been suggested using slightly different metrics. For example, Emmanuel Perry's WAR model [17] which relies heavily on quality over just shot quantity by using Corsica's xGF metric for shots [2]. Many models are no longer public due to their authors being hired by professional teams[1].

Dom Luszczyszyn created a model [12] which attributes a game score to each player. The game score is meant to be on a similar scale to points scored by the player during the game instead of being a value relative to replacement level players. The score is calculated as a weighted average of many in-game stats for each player. It values primary assists over secondary assists, takes into account penalties drawn and taken as well as the quantity of shots for and against while a player is on the ice. The idea is based off of the game score metric for basketball created by John Hollinger[26]. While the base game score value does not account for context or competition, Luszczyszyn has added an adjusted score metric that takes into the quality of teammates and competition in his final weighted scores [13].

2.3.2 Shot Quality Modelling

Some of the models above consider shot quality while others consider all shots equally. However, it is clear when examining shot data that not all shots are created equal [18]. As a result, many have developed different ways to assess the shot quality based on publicly available data. Earlier models assessed shot quality based on shot distance given different game situations and shot type. Alan Ryder's model [19] took into account whether a shot was a rebound, a long shot or a scramble shot. A rebound was defined as a shot taken no less than 2 seconds after another shot. A scramble shot was defined as a shot within 6 feet which is not a rebound. Ken Krzywicki built upon his model, adding the period during which the shot was taken to his model [9]. Corsica's newer shot quality model adds in shot angle to their calculation [2] and reduced the number of game context variables to rebounds and rush shots. A rush shot is defined to be a shot less than 5 seconds after an event occurred outside the offensive zone. Then they attribute a fraction of a goal to each shot. Aggregating these fractions, they estimate the expected number of goals to occur when a player is on the ice. They name this aggregate as the "expected goal for" metric or "xGF". It has more predictive power to similar models using just shot or shot attempts [16].

Chapter 3

TrueSkill with In-Game Events

TrueSkill was created for team competitions where players don't change throughout the game. e.g. Halo 2, tennis and to a lesser extent baseball and basketball. Furthermore, to distinguish between different player skills, TrueSkill needs players to play with a variety of teammates. TrueSkill struggles with franchises playing each other as it is hard to distinguish between the skill on individual players on each franchise. Here we define franchises as teams with a set group of players who only play for that franchise. So players will mostly play with the same team from game to game. An example in Halo of franchises would be squads that play all their games together. Here we are considering expanding TrueSkill to games where players on the field are often switched for other players on the team but the players on a team usually stay the same from game to game.

Some issues in adapting TrueSkill are that different players are no longer equality responsible for the result of the game. Secondly, the quality of competition that each player on a team sees can vary. Finally, players on the same team have the same game outcomes. This report suggests using in-game events to update player ratings instead of the final game outcome. Using in-game events will greatly increase the number of observations for each player, resulting in more accurately tracking the impact of a player on the game instead of using time played. The proposed model will also allow to distinguish between players on the same team based on the outcomes of when they were actively playing.

The goal of this model is to properly distinguish player ratings through in game events. We show in section 4.4 that this method leads to faster and more accurate detection of a player's skill.

3.1 In-Game Events

The goal is to take into account how the game went instead of a binary win or loss when updating player skill. The in-game events will better reflect a player's impact on the game. We label these in-game events as either scoring or non-scoring events. We define scoring events as events that directly contribute to a team winning or losing the game. A team wins the game by achieving more scoring events than their opponent. Some examples of scoring events in different games are getting a kill in a team-deathmatch game in Halo, capturing a flag in a Capture The Flag game in Halo, scoring a goal in hockey or soccer, and making a basket in basketball.

Conversely, non-scoring events do not directly contribute to winning the game. However, they can have a positive or negative impact on the game and can lead to a scoring event. Some examples of non-scoring events in different games are taking a shot in hockey or soccer, making a pass in basketball, picking up a strong weapon in a team-deathmatch in Halo, and getting a kill in a Capture The Flag game in Halo.

The proposed methodology will use both scoring events and non-scoring events. At each event, active players from both teams will be credited with the outcome of that event. Each non-scoring event will be weighted using the probability that it leads to a scoring event.

3.2 Weighting In-Game Events

The extend to which an event impacts the outcome of the game will dictate how much player ratings should be adjusted. For non-scoring events, we will calculate this as the probability that the event leads to a scoring event for either team. There are two different approaches we could take to calculate this weight.

The first approach is based off of the THoR rating system [20] used for hockey players. In the THoR rating system, each type of in-game event is given an NP20 score to determine it's value which is then attributed to the player performing the event. NP20 stands for net probability after 20 seconds. It gives the probability that a goal occurs in the 20 seconds following the event for the team for which the event occurred. The value is the probability that the team scores minus the probability that the other team scores. 20 seconds was chosen as the effect of events on the score seemed to no longer be significant afterwards. This approach looks to attribute the impact of a type of event based on how likely a scoring event occurs shortly afterwards. This could be used in other games like soccer or in Halo.

The length of time for which to determine the impact of the event would likely need to change from game to game. For example, a lower pace and lower scoring event game like soccer might need a larger window when assessing the impact of non-scoring events.

The second approach would instead look at the outcome of a possession within a game. This is based off of a paper looking at passing networks in basketball [25] to cluster players based on their style of play and performance. In their model, they constructed a network for each possession where there are a set number of outcomes. This approach would weight events based on the expected value of the outcome of the play given an event occurred. This method does have challenges where we need to properly weight the value of an event that could occur multiple times in the same play. This method makes more sense for cases like basketball where possession is clearly defined and delineated. Each game can be broken down into a series of possession plays, making it easier to evaluate events in this context.

Once we have assigned a weighting to the outcome of each event, all players active at the time of the event will have their skills updated as a result of the event. This will allow us to only adjust players for the events they can control. The player update will be scaled proportionally to the weight of the event. Therefore, players winning high-impact events will have greater rating increases than players winning low-impact events. The size of the update will be directly proportional the the weight of the event. Given an event weight, ω, the skill mean and variance update for a player will be

$$\mu_{new} = \mu_{old} + \omega \times v\frac{\sigma_{old}^2 + \tau^2}{c}$$
$$\sigma_{new}^2 = \left(\sigma_{old}^2 + \tau^2\right)\left(1 - |\omega| \times w\frac{\sigma_{old}^2 + \tau^2}{c^2}\right)$$

This is slightly different than the TrueSkill method outlined in section 2.2.1. We applied the weight ω scale to the mean scaling factor v as well as to the variance scaling factor w. For the variance, we use the absolute value of the weight since we want to do the same size update whether the update is positive or negative. One caveat of this approach is that there are no longer ties. All events are either a net positive or negative for a team. Any event that has no positive or negative impact on a scoring event game does not need to be considered.

3.3 Including Game Context

When observing specific game events, including the game context can help better understand the impact of certain players. There are many factors that can cause one team to

have an advantage over the other during a part of the game or during the whole game. If players achieve good results in advantaged situations, it makes sense to take that into account when updating ratings.

A game-long advantage could be a map in Halo where one team is advantaged over the other team due to the characteristic of the map. Another example is the home field advantage in many sports. It has been well documented that across all major sports, there is an advantage for the home team [15]. Non-scoring events might also favor the home team. This was noticed in hockey where these events are recorded by in-house statisticians [4].

An example of a temporary advantage that could impact a player's performance is a team being down a player due to a player disconnecting in Halo or due to a red card in soccer. Another example could be a team being on a power-play in hockey. These contexts could be useful to account for since they are not present for the entire game, only for certain events.

To calculate the weighting of these advantages, we follow a similar method to the one used by Lanfried [11] to calculate the skill of players on different playing surfaces. We will add a generic game context player to a team for each advantage. Each game context player c_i will start with mean $\mu_{c_i} = 0$ and with the same starting variance $\sigma_{c_i}^2 = \sigma_0^2$ as the players. The game context will be included in the team performance calculation as shown below. As they are not a true player, we will not add performance variance β to the game context but instead a dynamics variance of τ.

Assume that we have two sets of players $\{a_1, ...\} \in \mathcal{A}, \{b_1, ...\} \in \mathcal{B}$ representing our two teams. Team A has the set of m game contexts $\{c_1, ..., c_m\} \in \mathcal{C}_A$ for the given event. Team B has the set of n game contexts $\{c_{m+1}, ..., c_{m+n}\} \in \mathcal{C}_D$ for the given event. Δ and c are calculated as follows.

$$\Delta = \left(\sum_{a_i \in \mathcal{A}} \mu_{a_i} + \sum_{c_i \in \mathcal{C}_A} \mu_{c_i}\right) - \left(\sum_{b_i \in \mathcal{B}} \mu_{b_i} + \sum_{c_i \in \mathcal{C}_B} \mu_{c_i}\right)$$

$$c = \sqrt{\left(|\mathcal{A}| + |\mathcal{B}|\right)\beta^2 + \sum_{\alpha \in (\mathcal{A} \bigcup \mathcal{B} \bigcup \mathcal{C}_A \bigcup \mathcal{C}_B)} \sigma_\alpha^2}$$

The values of μ and σ are updated for all context elements and players after each event. Enough events need to be simulated for all individual context elements to reach a certain stability in their rating. Once that has occurred, the average value of μ is taken after it has reached that stability. The average μ will be used as a constant to represent the value of

that game context. We will assume it to be the true underlying value of the game context. When then re-run all events through our model, applying the weightings of the game context when calculating Δ as shown above and recalculate the player skills accounting for game context. As the context's no longer have any variance, c will be calculated as it was in section 3.2.

Chapter 4

Simulations

To asses the validity of the model we will simulate games between players with set initial ratings. The outcomes of those games are then passed into the model without the knowledge of the prior player ratings. The idea is that the model will properly detects each player's skill given a reasonable number of samples. If it can not extract the player skill ratings given these games, then it will be unable to detect player skills in a real world example.

4.1 Estimating Skill Using Game Outcomes

We will first verify the TrueSkill model can detect underlying player skill based on the outcomes of simulated games. This can give us an idea of how many samples the TrueSkill model needs to detect player skill as well as ensure our processes are correct.

First, we need to set the constants for our modeling system. We will set the starting player skill mean as $\mu_0 = 1$ and the rest of the parameters will be set using the suggested values in the original Trueskill paper [6]. The starting variance is set to $\sigma_0^2 = \left(\frac{\mu_0}{3}\right)^2 = \frac{1}{9}$, the player's performance variance is set to $\beta = \frac{\sigma_0^2}{4} = \frac{1}{36}$ and set the dynamics variance to $\tau = \left(\frac{\sigma_0}{100}\right)^2 = 1/90000$.

Then we create our set of players $\mathcal{A}$. Each player i is given a mean skill μ_i. Their skill variance will be equal to the dynamics variance τ of the system.

4.1.1 Single Player Contests

We will begin with single player contests. Examples of single player contests are a chess match, a singles tennis match or a billiards game. In this case, the TrueSkill model works identically to a Glicko model. To simulate the outcome of single player contests, we will start by taking two random opponents $i, j \in \mathcal{A}$. The opponents must be different players, so $i \neq j$. In the TrueSkill algorithm, each player's performance p_i follows a Gaussian distribution with the player's mean skill rating as the mean. The variance is given by the variance of the player's underlying skill, which in our case is just the dynamics variance τ, plus the generic performance variance β. So, we sample each player's performance p_i from the distributions $P_i \sim N(\mu_i, \tau + \beta)$.

If $p_i < p_j$, then player j has won the game and *vis versa*. Once the game outcome has been recorded, the process is repeated for the next game until the required number of games have been simulated. For our purposes, we will assume no ties can occur since there will be no ties when we simulate events.

The games are run through the TrueSkill model, without the knowledge of the players actual skill, to calculate each player's skill rating. Details on how to run the TrueSkill algorithm in this simple case is covered in section 2.2.1. Each player is given an initial rating of μ_0 and variance of σ_0^2. Since the TrueSkill model sets player rankings based on the relative ranking of other players, if you want to achieve the actual underlying values of μ_i for each player, then you will need the sum of starting mean ratings to be equal to the sum of underlying ratings $\sum_{i \in \mathcal{A}} \mu_i = \mu_0 \times |\mathcal{A}|$. Otherwise, the relative ratings of the players should tend to the underlying ratings, but the absolute ratings will not.

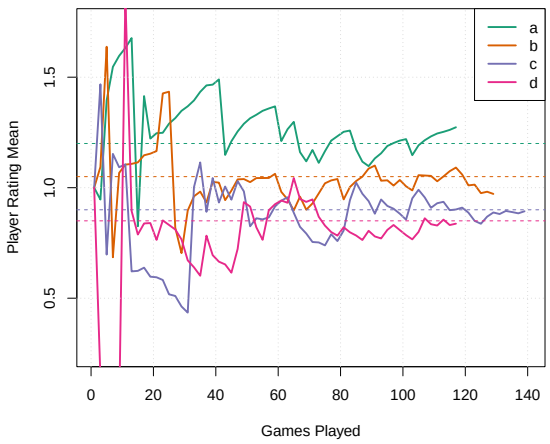


Figure 4.1: Single Player Contests

Figure 4.1 shows the results of single player contests simulated as described above between a set of four players $\{a, b, c, d\}$ where a pair of players chosen are at random over 300 games. The underlying player ratings used for simulating the games were $\mu_a = 1.2$, $\mu_b = 1.05$, $\mu_c = 0.9$, $\mu_d = 0.85$. The results of the games were then run through the TrueSkill model. All players started with $\mu_0 = 1$. After about 50 games, the μ rating

of each player had started to converge towards its underlying value, represented by the dotted line in Figure 4.1. The rating fluctuates over time as outcomes change, but remain close to the true underlying skill. This is the expected behavior we will want to remain constant as we modify the model. Note that the player's ratings only reach their proper values because the sum of their initial mean ratings is equal to their underlying ratings.

4.1.2 Multiplayer Contests without Franchises

Next we will simulate the outcome of multiplayer contests with k players on each team. These games will between random teams and not between franchises. This is like a multiplayer Halo match which is the exact case for which the TrueSkill model was originally created. The games are team based and the composition of the teams change per game. A player that was your opponent one game can just as easily be your teammate in the next game.

Let us build random teams of k players each. Start by sample without replacement $2 \times k$ players $\{a_1, ..., a_k, a_{k+1}, ..., a_{2k}\}$ from the set of players $\mathcal{A}$ in a random order. Assign the first k players $\{a_1, ..., a_k\}$ to team 1 and the remaining k players $\{a_{k+1}, ..., a_{2k}\}$ to team 2. We calculate the performance p_i of each player in the same way as shown in 4.1.1. The performance of the team is then calculated as the sum of each team member's individual performance.

$$p_1 = \sum_{i=1}^{k} p_i \qquad p_2 = \sum_{i=k+1}^{2k} p_i$$

If $p_1 < p_2$, then the players on team 1 have won the game and *vis versa*. We keep track of the players on both teams as well of the outcome. Once that is done, we repeat the process, choosing different random teams each time until the required number of games have been simulated.

Next, the games are run through the TrueSkill model. The details on how to run this model are covered in section 2.2.2.

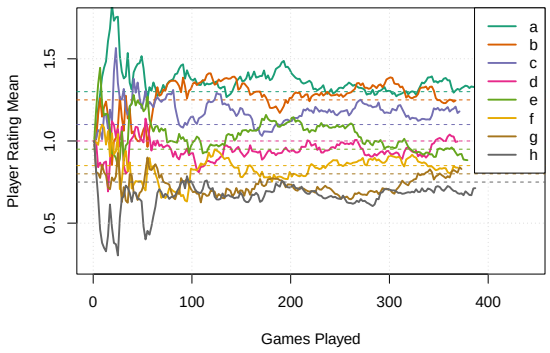


Figure 4.2: Multiplayer Contests

In Figure 4.2, we show the results of ranking 8 players in random multiplayer games with 3 person teams. The players were

labeled a to h, a being the strongest player and h being the weakest. Again, we can see the player ratings tending towards their underlying skill after around 80 to 100 observed games. It took a bit longer to reach the proper rating than in the single player contests, but not by a too large margin.

4.1.3 Multiplayer Contests with Franchises

The goal of our model is to extend the TrueSkill algorithm to work in cases where teams might be more static and player skill is harder to extract. Therefore, we will now evaluate the strength of the TrueSkill algorithm in cases where players belong to separate franchises $\mathcal{A}_i$ where a player can only belong to one franchise. If $a_1 \in \mathcal{A}_i$ then $a_1 \notin \mathcal{A}_j$ where $j \neq i$.

To simulate these contests, we first create n franchises $\mathcal{A}_1, ..., \mathcal{A}_n$ with at least k players in each. Then, we randomly select 2 franchises from $\{\mathcal{A}_1, ..., \mathcal{A}_n\}$ and randomly select k players from both those franchises. With the two teams of players, we then proceed the same way as described in 4.1.2 to simulate the outcome of the game. We continue this process, each time selecting 2 random franchises and k random players from those franchises until we have reached the desired number of games.

The outcomes of the games are run through the TrueSkill using the methodology outlined in 2.2.2.

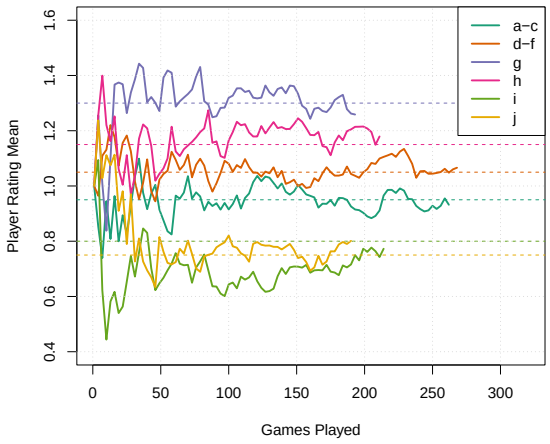


Figure 4.3: Multiplayer Contests with Teams

We simulated 3 person multiplayer games from 3 franchises: $\{a, b, c\}, \{d, e, f\}$ and $\{g, h, i, j\}$. The first two franchises have 3 players each, so all players play in every game. The last franchise has 4 players, so the makeup of the team will change from game to game. For the first two franchises, since all players play all the games, it is impossible to differentiate the between the skill of different players, giving them all have the same skill. The skill of players of those franchises will tend to the average skill of the franchise. The players on the last franchise will have different skill ratings based on how well the team does with and not without them. Their individual ratings will tend to their true value. In Figure 4.3, you can see that for franchises $\{a, b, c\}$ and $\{d, e, f\}$, where each player played every game, all player ratings stayed the same. In team $\{g, h, i, j\}$, where the players changed in each

game, the model was able to properly detect each player's underlying skill. This shows a limit of TrueSkill using game outcomes if most of the players on a franchise always play each game.

4.1.4 Contests with Game Context

Next, we will simulate the model described in section 2.2.4. In that section, we discussed work by Gustavo Landfried which consisted of creating additional players in tennis matchups based on the type of court the match was played on. This method is used to add game contexts to our model.

To recreate the tennis scenario, we will take a set of 3 players: $\{a, b, c\}$. Each game will be played between 2 randomly selected players from the set. For each game, we will also randomly select whether the game will be played on a clay or a grass court. Once the court type is selected, we will add a player of that court type to each team. For example, let player a be playing against player c on a clay court. The set of players on the two teams will be $\{a, a\text{-}Clay\}$ versus $\{c, c\text{-}Clay\}$. Once the teams are set, the games are simulated as in section 4.1.2. The outcomes of the games are then passed into the TrueSkill algorithm.

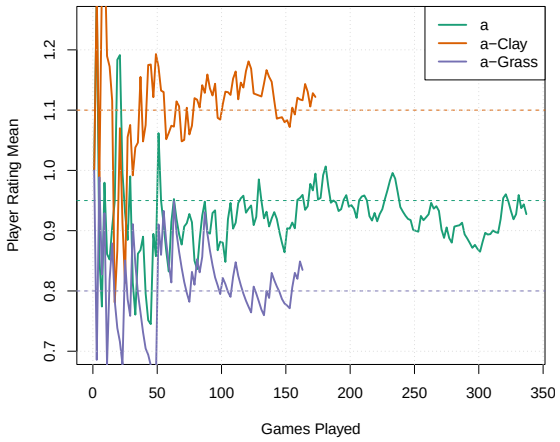


Figure 4.4: Single Player Tennis Contests

For our simulation, player a had mean skill $\mu_a = 0.95$ and is slightly stronger on clay with $\mu_{a\text{-}Clay} = 1.1$ and $\mu_{a\text{-}Grass} = 0.8$. Player b had consistent skill mean skill $\mu_b = \mu_{b\text{-}Clay} = \mu_{b\text{-}Grass} = 1.05$. Player c had mean skill $\mu_c = 1$ and is slightly stronger on grass with $\mu_{c\text{-}Clay} = 0.9$ and $\mu_{c\text{-}Grass} = 1.1$. In Figure 4.4, we can see that the algorithm is able to properly detect the true skill of player a as well as their strength of different types of courts. Therefore, we should be able to use the TrueSkill algorithm to detect the impact of different underlying game context.

4.2 Estimating Skill Using In-Game Events

Now to test through simulation whether using weighted outcomes can still allow us to detect the underlying player skill. While the number of events to detect the skill will likely increase, the hope is that by seeing more events per game will skill allow us to quickly detect player skill. This paper discussed some player value models discussed in section 2.3.1 that showed different ways to weight in-game events for hockey. We will use some of these methods as a basis for how our events are weighted.

Using the original values of β and σ_0^2, the model using in-game events was slow at converging to the underlying ratings. To increase the speed of the algorithm, the initial parameters were changed to $\sigma_0^2 = \left(\frac{1}{8}\right)^2$ and $\beta = \frac{\sigma_0^2}{2}$. The reason behind the parameter adjustments are discussed further in section 4.3.

4.2.1 Weighted Individual Events

We will start by simulating events the same way we were simulating games in section 4.1, except each outcome will be assigned a random weight. For this chapter, we used hockey events to decide on the weight distribution. The hockey weights were calculated using a mix of the ThoR model and the xGF shot quality model. More details on how the hockey weights were calculated can be found in section 5.2. Note that any method is valid to generate the random weights ω_i as long as $0 < \omega_i \leq 1$. Figure 4.5 shows the weight distribution of hockey events against an exponential distribution with rate 30. We found that the distribution was close enough to an exponential distribution for the sake of our simulations. Therefore, for each simulated event i, it will be given the weight ω_i from the distribution $\Omega \sim Exp(30)$. If $\omega_i > 1$, we set $\omega_i = 1$ but note that $P(\Omega > 1) = e^{-30} \approx 9.3 \times 10^{-14}$. Therefore, this will not significantly change the distribution curve shown in Figure 4.5.

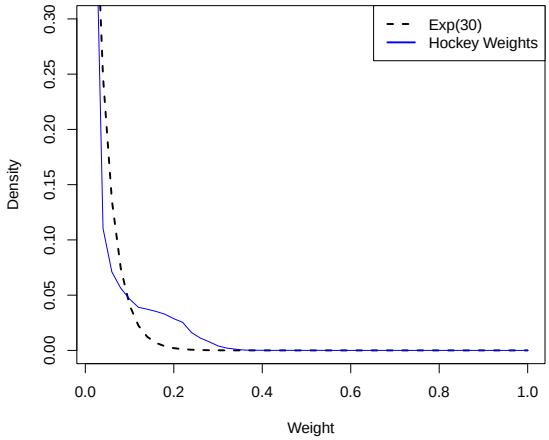


Figure 4.5: Hockey Event Weights

These weighted events are then used in the model using the methodology shown in section 3.2 to update player ratings.

In Figure 4.6, you can see players rated from events with random weights. As a result of the smaller weightings for each individual event, player ratings move much less sporadically, especially in the first 10 or so events. However, after 600 or so events, TrueSkill was able to detect the reach the underlying rating. Comparing Figure 4.6 to Figure 4.2, the player ratings are more stable when using weighted events. However, the absolute number of events to reach the proper rating is larger than the absolute number of games. If there are many events per game per player, this method could still be significantly faster than just looking at game outcomes.

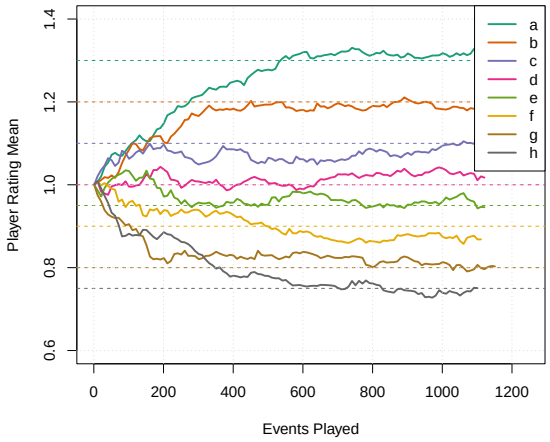


Figure 4.6: Weighted Multiplayer Contests

4.2.2 Adding Game Context to Events

As discussed in section 3.3 and shown in section 4.1.4, we can add game context to modify a team's expected performance. This is to account for advantages that players may have that might be outside of their control and for which they should not get credit. To achieve this, when simulating events, we will randomly attribute a context c_i to a team. The other team is attributed the opposite context c_{-i}. Each context c_i has a mean rating μ_{c_i} and a variance equal to the dynamics variance τ. The context performance is sampled as $P_{c_i} \sim N(\mu_{c_i}, \tau)$ and $P_{c_{-i}} \sim N(-\mu_{c_i}, \tau)$.

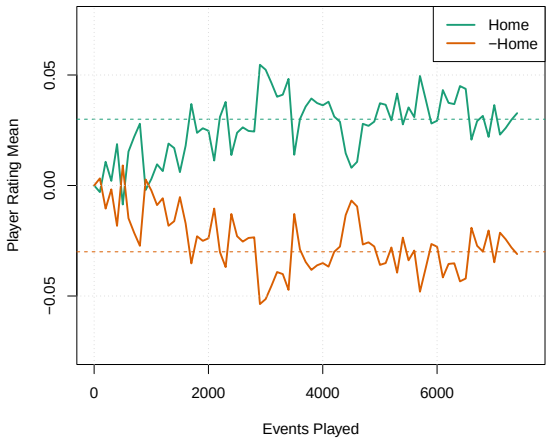


Figure 4.7: Multiplayer Contests Contexts

Once the context's performance is sampled, it is added to each team's performance. We then proceed as in section 4.2.1 to detect the winner and weight of each event. Once all events are simulated, we run the model as outlined in section 3.3. Once all events are run through the model, we can observe the results for different contexts.

We set a home advantage of $\mu = 0.03$ in our simulations. In Figure 4.7 can see that the model properly detected the advantage of being home and the disadvantage of not being home. In Figure 4.7, we can see that the ratings seem to stabilize after the first 2000 events played. Taking the average of the remaining values of μ gives us a value of $\mu_{home} = 0.0318$ which is pretty close to the underlying value of 0.03. Once the estimate for μ_{home} is calculated, we can rerun the events in the model with a fixed context value of μ_{home} to get a better player rating estimate.

4.2.3 High Substitution Contests

A limitation of using TrueSkill on outcomes was for games where many player substitutions occur during the game. The hope is that using TrueSkill on event outcomes will improve the detection of player skill. To test this, we will base our simulations off of hockey. In a hockey game, players will stay on the ice for very short periods of time, called shifts. A typical shift in hockey last about 45 seconds, after which they change and new players come on [24]. This means that each player will only play a fraction of the complete game and may play against a wide range of players throughout the game. Therefore, calculating each player's skill solely on the game's outcome may not be an efficient way to detect their skill since they only have a limited impact on the outcome of the game. However, looking only at in-game events where the playing is actively playing should give us a better picture of their skill.

Each team will be comprised of the typical hockey formation of 18 players $\{a_1, ..., a_{18}\}$. Our simulations will be ignoring goalies since they effect play very differently than the rest of players. In hockey, players will play with their lines for most of the game. A team has 4 forward lines with 3 players each and 3 defense lines with 2 players each. To mimic this, the players will be split into 4 distinct sets $\{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4\}$ of 3 players and into 3 distinct sets $\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$ of 2 players. We will assign the players to each set as such:

$$\begin{aligned} a_1, a_2, a_3 &\in \mathcal{F}_1 & \qquad a_{13}, a_{14} &\in \mathcal{D}_1 \\ a_4, a_5, a_6 &\in \mathcal{F}_2 & \qquad a_{15}, a_{16} &\in \mathcal{D}_2 \\ a_7, a_8, a_9 &\in \mathcal{F}_3 & \qquad a_{17}, a_{18} &\in \mathcal{D}_3 \\ a_{10}, a_{11}, a_{12} &\in \mathcal{F}_4 \end{aligned}$$

On a regular hockey team, each line will not be equal. Some lines will be stronger than others and as a result will receive more ice time. To take this into account in our simulation, when an event occurs and we will favor certain sets of players when deciding

which line was playing. We will assume that in terms of general strength, the lines are ordered as $\mathcal{F}_1 > \mathcal{F}_2 > \mathcal{F}_3 > \mathcal{F}_4$ and $\mathcal{D}_1 > \mathcal{D}_2 > \mathcal{D}_3$. We will assume each line has the following amount of ice team in a regular 60 minute game. From that we can infer how likely a forward player f or a defenseman d from that line is playing at any given moment.

$$\begin{aligned}
\mathcal{F}_1 &: 18 \text{ mins } \rightarrow P(f \in \mathcal{F}_1) = 30\% & \mathcal{D}_1 &: 22 \text{ mins } \rightarrow P(d \in \mathcal{D}_1) = 36.7\% \\
\mathcal{F}_2 &: 16 \text{ mins } \rightarrow P(f \in \mathcal{F}_2) = 26.7\% & \mathcal{D}_2 &: 20 \text{ mins } \rightarrow P(d \in \mathcal{D}_2) = 33.3\% \\
\mathcal{F}_3 &: 14 \text{ mins } \rightarrow P(f \in \mathcal{F}_3) = 23.3\% & \mathcal{D}_3 &: 18 \text{ mins } \rightarrow P(d \in \mathcal{D}_3) = 30\% \\
\mathcal{F}_4 &: 12 \text{ mins } \rightarrow P(f \in \mathcal{F}_4) = 20\%
\end{aligned}$$

For each event of the game, we will randomly select the forward $\{\mathcal{F}_i\}_{i\in[1,4]}$ and defense $\{\mathcal{D}_j\}_{j\in[1,3]}$ lines that are on the ice using the probabilities above. All players $\{a_k\}_{k\in[1,18]} \in \mathcal{F}_i \cup \mathcal{D}_j$ will make the active players for that event. We repeat this process to select random lines for the other team. We could also assume that the other team matches their opponents lines, *i.e.* if the first team has lines $\mathcal{F}_2, \mathcal{D}_3$ on the ice, then the second team will have lines $\mathcal{F}_2, \mathcal{D}_3$ as well. This is a strategy sometimes used by coaches in hockey. We don't do that in this simulation, but that is a case which could arise in a real-world example that would reduce the effectiveness of the algorithm.

Finally, we need to decide how many events will occur over the course of the game. Looking over NHL games from the 2018-2019 season to the 2021-2022 season, the number of events per game has a bell curve shape with a mean of 200 and a standard deviation of 25. Therefore, we will randomly set the number of events e per games by sampling from $E \sim N(200, 25)$.

To simulate a hockey game, simulate e events. For each event, randomly select the lines from each team to determine the players on the ice. Then calculate the weight of the event from $\Omega \sim Exp(30)$. Once both teams are set, we randomly sample their performance as done previously and detect which team won the event by comparing the performance of both teams. The outcome is recorded and we continue to the next event.

Team	Line	Actual Skill	Model Skill	Diff
A	F1	1.3	1.27	-0.03
A	F2	1.25	1.19	-0.06
A	F3	0.95	0.82	-0.13
A	F4	0.8	0.67	-0.13
A	D1	1.05	1.17	0.12
A	D2	0.85	0.96	0.11
A	D3	0.75	0.84	0.09
B	F1	1.1	1.21	0.11
B	F2	0.95	1.05	0.1
B	F3	0.85	0.93	0.08
B	F4	0.8	0.83	0.03
B	D1	1.25	1.16	-0.09
B	D2	1.2	1.09	-0.11
B	D3	0.9	0.75	-0.15

Table 4.1: Hockey Game Simulations

This is done until all events in the game
have been recorded and we move on to the next game.

There are many simplifications in this method of simulating hockey contests. These were left out in the goal of simplicity. Further work could be done to improve this framework.

- Power plays are ignored
- Players are not kept on the ice for continuous shifts
- Events are not differentiated
- Previous events have no impact on which events might come next

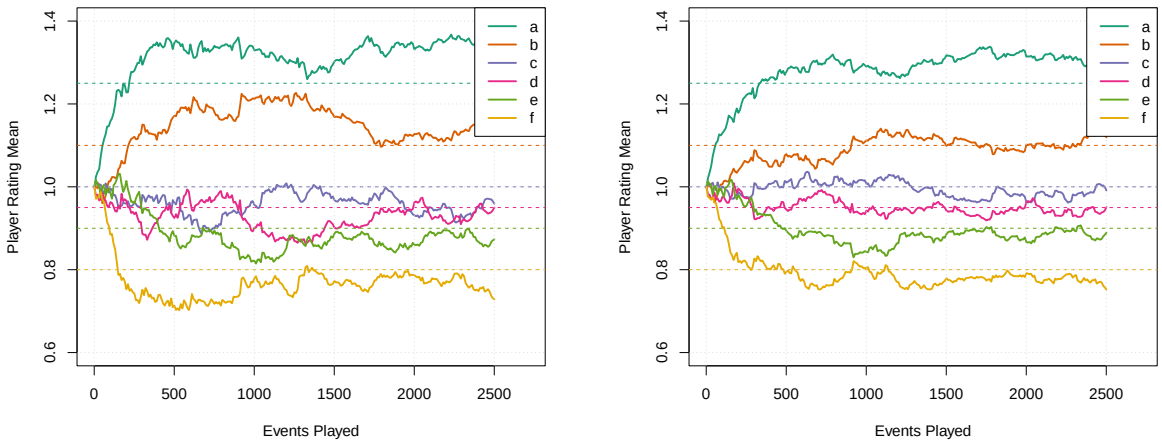
In Table 4.1, we see the results of simulating two hockey teams against each other over 50 games of events. We did not match lines against one another. Team A has a strong offense, especially it's first two lines. Team A's defensive lines are weaker. Team B is the opposite with a stronger defense core and weaker offensive lines. We can see that in both cases, the stronger of the offense or the defense for a team was underrated by the model and the weaker of the two was overrated by the model. This is due to the fact that the model can't fully distinguish between the relative strength of groups of players on the ice. This shows a weakness in the model where the players on the ice is not fully random. You will always see some forwards and some defensemen on the ice. If all defensemen on a team a much weaker, it will be harder for a forward to win as many events as he maybe should. However, the model did do a great job of ranking similar type players within a team. When analysing real data, taking into account these limitations will be important.

4.3 Parameter Tuning

While running event simulations, it was observed that player ratings would often spread out too much after the first few event outcomes. Only after many more events would the spread of player skill contract again and tend towards the true underlying skill levels. Increasing the beta and decreasing the starting variance reduced this phenomenon.

In Figures 4.8a and 4.8b, we see the results of the TrueSkill algorithm ratings players based on events taken from 2 player multiplayer matches. The first figure uses the basic parameters outlined at the start of section 4.1; $\sigma_0^2 = \left(\frac{1}{3}\right)^2$ and $\beta = \frac{\sigma_0^2}{4}$. The second figure

uses values used starting in section 4.2.1: $\sigma_0^2 = \left(\frac{1}{8}\right)^2$ and $\beta = \frac{\sigma_0^2}{2}$. The player mean ratings reach their true value a bit slower in Figure 4.8b, but mostly stop increasing or decreasing once they have reached the theoretical value.



(a) Base Parameters

(b) Updated Parameters

The beta is double in value relative to the variance compared to the suggested value in the TrueSkill paper. Having a larger beta makes sense when evaluating events since the variance of a player's performance will be larger on a single event than from a whole game. The lower starting variance is to counter the fact that we are decreasing variance much more slowly with weighted events.

4.4 Comparing In-Game Events to Game Outcomes

To see the efficiency of using events to rate players rather than game outcomes, let's simulate game outcomes based on events. We will assume players for each team play the whole game and are involved in all events. Otherwise, the event-based modeling will always be more accurate than the game-based modeling since it can distinguish between players.

We will simulate 50 events per game. Each event will be given a random weight following the same method outlined in section 4.2.1. After each event with a weight ω_i, we will randomly determine whether a team has scored a point. Say team A won event i with weight ω_i. We then detect whether team A or team B has scored a point by checking the following conditions. Team A scored a point if $\omega_i > 0$ and $u \leq \omega_i$ where $U \sim U(0, 1)$. Team B scored a point if $\omega_i < 0$ and $u \leq |\omega_i|$ where $U \sim U(0, 1)$.

Repeat this process for each event in the game and compare the points for each team at the end. The team with the most points is recorded as having won the game. If the game is tied, continue simulating new events until one of the teams scores a point. Once a team has scored a point, they have won the game.

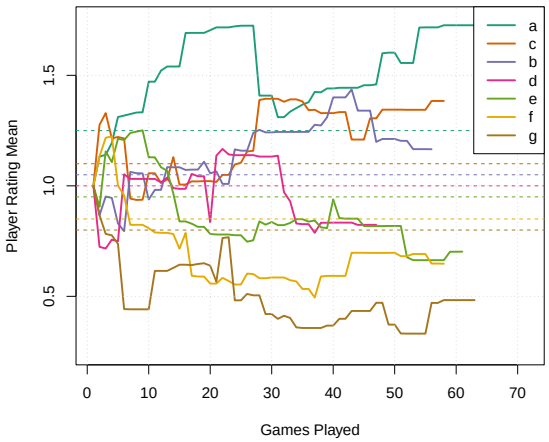


Figure 4.9: Ranking Game Outcomes

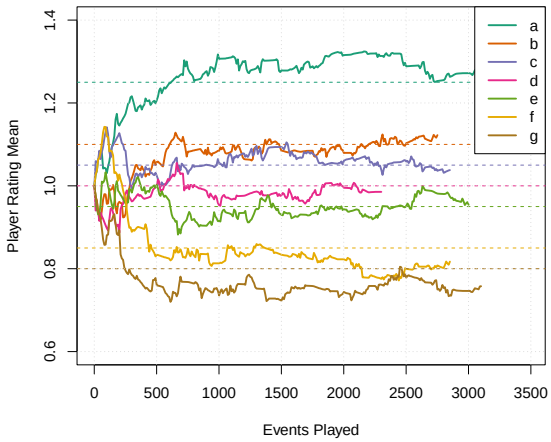


Figure 4.10: Ranking Events

In Figures 4.9 and 4.10, we can compare how well the TrueSkill algorithm does ranking players using game outcomes versus using events. In this example, we simulated 100 2 player multiplayer games. Each game, the teams were randomly created using a set of 7 players. The event weighting was using the weighting discussed in section 4.2.1. We can see the events based model offers a much more stable and accurate player ratings. This makes sense since they sample count is greatly larger than the games played.

Chapter 5

Hockey Implementation

This report will use hockey as an implementation event-based TrueSkill. Hockey is a very fluid game with possession of the puck changing sides many times during play. The data used for this paper was obtained from the NHL public API. The API endpoints used is documented in appendix A. Given the granularity of the data and the style of play in hockey, the first method of weighting events described in section 3.2 will be used.

5.1 Implementing Event Based TrueSkill in Hockey

Players will be ranked using on-ice events from the 2018-2019 season to the 2021-2022 season. The event data is taken from the NHL API detailed in appendix A.1.4.

The considered on-ice events are goals, shots, missed shots, blocked shots, giveaways, takeaways, hits, faceoffs and penalties. The data also contains stoppage in play events. Stoppages in play will be used to calculate certain metrics, but does not have a location associated to it in the data and won't be considered as an event for player ratings.

Overtime events were omitted as skill on 3-3 play might be very different than regular play. Goalies were also excluded from the analysis since they have a very different impact on the flow of play than the rest of skaters.

Each event will be given a weight representing the likelihood the event leads to a goal. Shifts data is used to determine which players were on the ice at the time, with the exception of faceoffs. The outcome of a faceoff mainly depends on the two players taking the faceoff, so that event is of little interest to us to rank all players on the ice. However, generating

faceoffs in advantageous locations can be a benefit. If a player is creating many faceoffs in the opponent's zone, it is likely due to good play and leads to good opportunities for their team. The inverse is true for creating many faceoffs in the defensive zone. Therefore, the creation of a faceoff will be considered as an event for the players on the ice at the time the faceoff was called.

All players will start with a mean rating of $\mu_0 = 100$. The remaining parameters will have the same relative values as discussed in section 4.3: $\sigma_0^2 = \left(\frac{100}{8}\right)^2 = 12.5^2$, $\beta = \frac{\sigma_0^2}{2} = \frac{12.5^2}{2}$ and $\tau = \left(\frac{\sigma_0}{100}\right)^2 = 0.125^2$.

5.2 Event Weighting

The weight of an event will be the probability of the event leading to a goal for the player's team. All events will be weighted on the expected impact and not the actual impact of that event. Therefore, goals will have weights equal to the probability that the goal would have occurred.

Most events will be weighted using the NP20 method from the THoR hockey model discussed in section 3.2. The zone in which the event happened will be tracked when calculating the NP20 value for each type of event. An event either occurs in the offensive zone, neutral zone or defensive zone. The NP20 value in zone i for an event is calculated as $\frac{\text{goals within 20s of event in zone i}}{\text{event in zone i}}$. Goals that occur after any stoppage in play within those 20 seconds are ignored. Goals for the team are counted as a positive and goals against are counted as a negative. The result gives us the expected number of goals for in the 20 seconds following an event in a particular zone. Below are examples of the calculated NP20 value for a few event types.

Event Type	Offensive	Neutral	Defensive
Giveaway	-0.0146	-0.0254	-0.0434
Hit	0.0046	-0.0025	-0.0105
Missed Shot	0.0169	0.0202	0.0296

Table 5.1: Event NP20 Values By Zone

Blocked Shots, **hits**, **giveaways**, and **takeaways** are weighted using purely their NP20 value. The quality of the shot need to be accounted for in the weight of shot related events.

To calculate the expected goal per shot, or "xG", Corsica's shot quality model [2] will be used. The probability of a goal p on a given shot is calculated using logistic regression on the shot type γ, the distance δ, the angle α, whether the shot is a rebound ρ, whether the shot is on the rush σ and whether the shot was from outside the offensive zone θ. The probability of a goal scored from outside the offensive zone is calculated as a flat probability since those goals are rare. Looking over all shot attempts from the last 4 seasons, 0.78% of the shots resulted in a goal. The probability P of a goal being scored on a shot given these parameters is then calculated by:

$$t = \beta_0^\gamma + \beta_1^\gamma \delta^3 + \beta_2^\gamma \delta^2 + \beta_3^\gamma \delta + \beta_4^\gamma \alpha^3 + \beta_5^\gamma \alpha^2 + \beta_6^\gamma \alpha + \beta_7^\gamma \rho + \beta_8^\gamma \sigma$$

$$P = \begin{cases} \theta = 1 : 0.0078 \\ \theta = 0 : \ \frac{1}{1+e^{-t}} \end{cases}$$

Figure 5.1 shows the calculated xG values for a snapshot taken on a rebound. The center of the net is located at coordinates (0,0). The end of the offensive zone is located at $X = 68$.

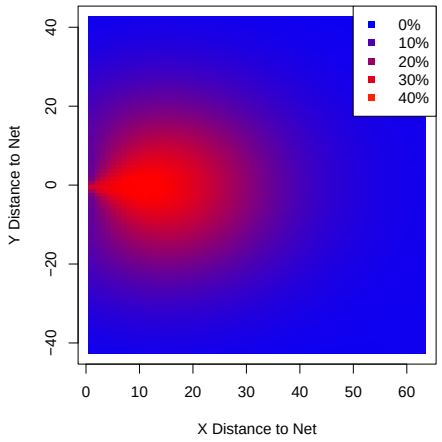


Figure 5.1: xG of Snapshot Rebound

The weight for **goals**, **shots** and **missed shots** will be the $\text{xG} + (1 - \text{xG}) \times \text{NP20}$. So the weight is the probability that the shot directly resulted in a goal plus the chance that a goal occurred shortly after, assuming the shot did not result in a goal. Goals will be given the NP20 value for shots since the NP20 value for a goal is always 0 as it creates a stoppage in play.

The league average power play percentage over the four observed seasons will be used to weight **penalties**. Over that time, teams scored on 20% of their power plays, so the weight will be 0.2.

For **faceoffs** events, the weight will be set using the NP20 value assuming either team has a 50% chance to win the faceoff. For a faceoff in the neutral zone, the weight will be 0 since the positive impact of winning a neutral zone faceoff is equal to the impact of the opponent winning a neutral zone faceoff. For a generated offensive zone faceoff, the weight will be equal to 50% times the NP20 value of a team winning a offensive zone faceoff plus 50% time the NP20 value of a team winning a defensive zone faceoff. The negative value of the result will give the weight for a defensive zone faceoff.

5.3 Game Context

Three different game contexts will be considered to add to the model. Home ice advantage, man advantages and offensive zone starts may impact the outcome on the ice and will be investigated further in this section.

5.3.1 Home Team Advantage

As discussed earlier in this report, it is an observed phenomenon that home teams tend to perform slightly better than their opponents in major sport leagues. Therefore, the "home" context will be added to the home team in all events and the "away" context will be added for the opposing team. Running the model over all 4 seasons, the means of the contexts reach a certain stability. The mean observed rating for the contexts are 2.67 for "home" and -2.67 for "away". There does seem to be a small advantage for home teams, as expected. The contexts weighted as $\mu_{\text{home}} = 2.67$ and $\mu_{\text{away}} = -2.67$ will be added to the model.

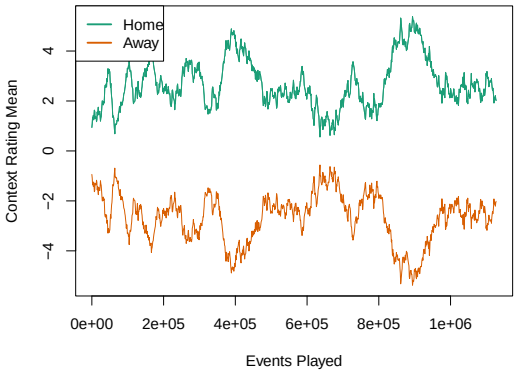


Figure 5.2: 2018 Home Ice Advantage

5.3.2 Man Advantages

In hockey, there are multiple ways for the teams to have a different number of players on the ice. The first is when a player performs an infraction, they are assessed a penalty, usually for 2 or 5 minutes. During the duration of the penalty, their team must play with one player less on the ice. This is called a power play. Teams will also elect to pull their goalie to put on an extra skater when the game has a couple minutes left and they need a goal to tie the game. Whenever teams have a different number of players on the ice at the time of an event, a man advantage context will be added. For example, if one team has 5 players and the other team has 4, the team with more players will be given the "5-4" context and the other team will be given the "4-5" context.

The only man advantages that are possible during a hockey game are "5-4", "5-3", "4-3", "6-5", "6-4", "6-3". Other man advantages seen in the data will be ignored as it is

likely due to bad shift data with too many or too few players recorded on the ice. If the shift data is incorrect, there is no way to know who was on the ice for the event. These bad data cases account for only 0.2% of all events.

To properly evaluate the impact of man advantages, instead of comparing the sum of team skill on the ice, the average team skill will be compared. The man advantage context will then account for the entirety of the difference between the two teams. The average team player skill will be scaled to the size of a normal team size to get a reasonable value for Δ. Let sets $\mathcal{A}, \mathcal{B}$ be the sets of the two teams of players on the ice in a man advantage situation where team $\mathcal{A}$ has won the event. The calculation of Δ is as follows.

$$\Delta = 5 \times \left(\frac{1}{|\mathcal{A}|} \sum_{\{i \in \mathcal{A}\}} \mu_i - \frac{1}{|\mathcal{B}|} \sum_{\{j \in \mathcal{B}\}} \mu_j \right)$$

The contexts with the largest sample size were the "5-4" and "4-5" contexts. Starting with $\mu_0 = 0$, they did see enough samples to t reach any stability in their rating. Therefore, the process was rerun, starting them at a higher value initial value, for example $\mu_0 = 20$, did see the context means reach a certain stability at around $\mu_{5\text{-}4} = 70$ and $\mu_{4\text{-}5} = -70$. This is close to but not quite the value of an average player given our starting rating for players of 100. So the disadvantage of being a man short is quite large.

An issue with how our model views the man advantage is that it will reward players whose play is more likely to lead to goals for their team. However, our model does not account as much for simply stifling the offense of the other team. When a team has a man disadvantage, the main goal is to reduce the number of scoring chances from the opponent. This is not something our model can account for. A player is not rewarded for the absence of events occurring on the ice. A way to incorporate this into the model would be to include ties. A tied event would be one where nothing has occurred on the ice for a certain amount of time, say 5 seconds. Players would be rewarded for achieving a tie when they are at a severe disadvantage. While this is true of all situations, it feels particularly important in a situation where such an imbalance is present. Therefore, it may be more productive to exclude man advantages from our current model and only compare events at even strength to better analyze player skill.

5.3.3 Offensive Zone Start

Starting a shift with an offensive zone draw should give those players a small advantage. They begin in a favorable position and could result in more positive events occurring on the

ice. Adding a context element for starting in the offensive zone could account for players mostly starting in offensive or defensive roles. When comparing the impact on an attacking team when taking a faceoff in the offensive zone, the weight of events in the following 10 seconds is quite positive as seen in Figure 5.3.

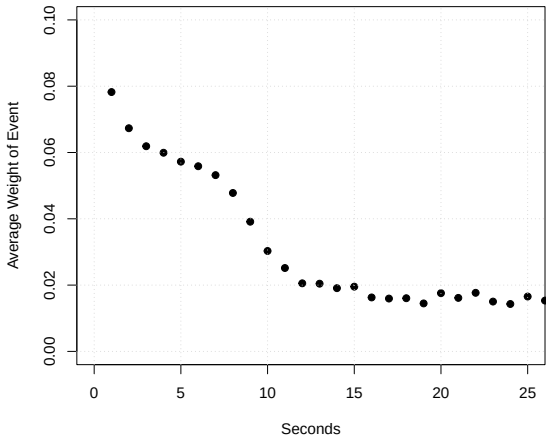


Figure 5.3: Offensive Zone Advantage

As expected, the large number of events seen were positive for the team with the "Offensive Zone Start" context. Creating "Offensive Zone Start" and "Defensive Zone Start" contexts for any event occurring within 10 seconds of a faceoff in the offensive zone created contexts that would continue to diverge in skill. Rerunning the model using much higher initial starting values for the contexts would not prevent it from continuing to diverge. The outcomes of events shortly after a faceoff are too positive to be effectively modeled. What should be rewarded is efficiently generating scoring chances from an offensive zone start. However, the model does not penalize the absence of events.te For that reason, this conxt will not be included to generate the results in section 5.4.

5.4 Results

The best and worst players when running the model on a single season can be seen in Table 5.3. While many of the top and bottom players change from year to year, there are some recurring names. Matthews has steadily climbed the rankings each year to end at the top in the 2021-2022 season. McDavid, considered to be the best active player in the NHL, tops the rankings once and only appears in the top 10 in the last two years.

There is also a tendency to see players from the same team for any given year. In 2018, Williams and Aho are both on Carolina, Theodore and Merrill play together in Vegas and Dunn and Tarasenko play for St Louis. In 2019, there are 5 Vegas players: Theodore, Pacioretty, Mark Stone, Holden and Stasny. Hamilton and Slavin both play defense for Carolina and Danault and Tatar play on the same line in Montreal. In 2020, Colorado has many players present with Devon Toews, MacKinnon, Rantanen, Makar, Girard and Landeskog. There is also McDavid and Puljujarvi from Edmonton. In 2021, Toronto's top

line of Matthews, Marner and Bunting are all present as well as many Boston players with McAvoy, Bergeron, Marchand and Pastrnak.

2018-2019		2019-2020		2020-2021		2021-2022	
Player	$\mu - \mu_0$	Player	$\mu - \mu_0$	Player	$\mu - \mu_0$	Player	$\mu - \mu_0$
Williams	2.14	Theodore	2.37	McDavid	1.91	Matthews	2.77
S. Aho	1.89	Pacioretty	1.97	D. Toews	1.82	McAvoy	2.54
Theodore	1.87	J. Slavin	1.74	MacKinnon	1.71	Bergeron	2.52
Merrill	1.75	M. Stone	1.69	Matthews	1.67	McDavid	2.47
Dillon	1.73	Holden	1.53	Rantanen	1.64	Weegar	2.27
Crosby	1.71	Danault	1.44	Puljujarvi	1.54	Marner	2.2
Dunn	1.65	Stastny	1.4	Makar	1.48	Bunting	2.15
Panarin	1.63	Matthews	1.39	Girard	1.44	Marchand	2.07
Tatar	1.54	Hamilton	1.36	Yandle	1.39	Pastrnak	2.02
Tarasenko	1.5	Tatar	1.32	Landeskog	1.33	M. Tkachuk	1.96
...	...	...	...	...	...	...	...
Pionk	-1.43	Khaira	-1.13	Gavrikov	-1.01	Reaves	-1.52
Manson	-1.44	Holzer	-1.18	Deslauriers	-1.04	Gambrell	-1.56
Ceci	-1.48	Bowey	-1.3	Eakin	-1.1	Peeke	-1.58
Laine	-1.49	Rowney	-1.32	T. Myers	-1.12	Galchenyuk	-1.63
Abdelkader	-1.49	Zaitsev	-1.32	Ristolainen	-1.16	Gavrikov	-1.65
M. Staal	-1.5	Grant	-1.34	Zadorov	-1.19	Gostisbehere	-1.69
Hagg	-1.61	Beagle	-1.35	Edler	-1.22	McCabe	-1.7
Doughty	-1.62	Filppula	-1.38	N. Schmidt	-1.35	Barbashev	-1.71
Sobotka	-1.84	Watson	-1.45	M. Staal	-1.36	Stralman	-1.87
Glendening	-2.14	Glendening	-1.56	Grant	-1.38	Mayo	-2.44

Table 5.2: Single Season Mean Rating

Notice that the player mean ratings do not differ much from their initial values. The events from the season did not have enough weight to cause proper separation between players. The player ratings can not tell us much about the probability of outcomes on the ice since any match-up of players will still be close to a 50/50. This means the ratings are not taking into account the quality of competition or of teammates if all players have essentially the same rating. A larger sample size to rate these players is needed. The highest and lowest rated players when running the model all 4 seasons consecutively are shown in Table 5.3. Some superstar forwards as well as elite young defenseman appear

at the top of the list. One surprise is Danault who is seen as a strong two way forward who is often charged with shutting down the other team's best players. The lowest rated player are mostly comprised of depth players. The one exception in the list is Laine who is a young elite sniper.

The separation between the best and worst players remains small though. Let us play the five best players against the five worst players and see the advantage for the stronger team. The two teams are $\mathcal{A}$ = {Matthews, Bergeron, Marchand, Theodore, McAvoy} and $\mathcal{B}$ = {Watson, Laine, Beagle, M. Staal, Glendening}. The performance P of team $\mathcal{A}$ is the sum of each individual performance.

$$P_{\mathcal{A}} \sim N\left(\sum_{\{i\in\mathcal{A}\}} \mu_i, |\mathcal{A}|\beta + \sum_{\{i\in\mathcal{A}\}} \sigma_i^2\right)$$

Best		Worst	
Player	$\mu - \mu_0$	Players	$\mu - \mu_0$
Matthews	5.51	Glendening	-5.08
Theodore	5.21	M. Staal	-4.06
McAvoy	4.72	Beagle	-3.89
Bergeron	4.66	Laine	-3.61
Marchand	4.61	Watson	-3.49
McDavid	4.37	Tierney	-3.13
Danault	4.15	Deslauriers	-3.03
Pastrnak	4.15	Kuraly	-2.97
D. Toews	4.14	Zaitsev	-2.95
MacKinnon	4.01	Khaira	-2.93

Table 5.3: All Seasons Mean Rating

The random variable $P_{\mathcal{A}} - P_{\mathcal{B}}$ can be derived to calculate the probability of $P_{\mathcal{A}} - P_{\mathcal{B}} > 0$. This random variable gives us the probability that team $\mathcal{A}$ would score the next goal against team $\mathcal{B}$.

The teams performance variables are calculated as $P_{\mathcal{A}} \sim N(524, 2460)$ and $P_{\mathcal{B}} \sim N(479, 2519)$. From those, the probability that team $\mathcal{A}$ would score the next goal against team $\mathcal{B}$ is calculated to be $P\left(P_{\mathcal{A}} - P_{\mathcal{B}} > 0\right) = 0.737$. This is not a huge disparity between the worst and best players in the league. This is to be expected since as player means had not converged to any value, especially those with higher or lower ratings. So the model had not finished estimating player skill.

Another option would be to rerun the events from a given year multiple times to get a better separation between players. The year's events should be replayed in the modeluntil there is convergence in most player ratings. Table 5.4 has results for player mean ratings after the 2021-2022 season events were run through the model 100 times when most player ratings had begun to stabilize.

Different names appear in the top 10 and bottom 10. McDavid retakes the top spot while Matthews appears a bit lower on the list. There are also a few lesser known names among the top players. Marchment jumps to second place after a great season in Florida that was cut short due to injury. Garland and Milano are seen as decent players but end up with surprisingly high rankings. Raffl is very surprising as he played as a depth player for Dallas. He is the only player in the top 10 that had, at one point, a mean rating below 100. After a few iterations through the season, his rating started to climb dramatically. This is an example of the quality of teammates and competition changing a player's rating.

Best		Worst	
Player	$\mu - \mu_0$	Players	$\mu - \mu_0$
McDavid	38.26	Weatherby	-41.26
Marchment	36.86	Gambrell	-39.52
J. Robertson	36.04	Blidh	-37.39
M. Tkachuk	33.16	Reaves	-36.66
Meier	29.38	Mayo	-36.28
Garland	29.21	Cizikas	-36.23
J. Slavin	28.21	Pederson	-32.93
Matthews	28.13	J. Benn	-32.52
Raffl	28.08	Beagle	-31.6
Milano	27.37	Richardson	-30.52

Table 5.4: 2021-2022 Modeled 100 Times

Next, let's look at all players from a single team over the 2021-2022 season. Calgary had very few injuries and therefore had a pretty stable lineup, which makes then a good team to look at. In Table 5.5, you can see the player ranking for all players which played at least 65 games for Calgary in 2021-2022.

Player	$\mu - \mu_0$
M. Tkachuk	33.16
Mangiapane	21.99
Dube	17.98
Zadorov	15.89
Coleman	13.16
Backlund	8.71
Monahan	7.07
Kylington	5.53
Hanifin	4.86
Lucic	2.94
R. Andersson	1.65
Gudbranson	0.53
J. Gaudreau	0.12
E. Lindholm	-0.14
C. Tanev	-9.46
Lewis	-12.24

Table 5.5: Calgary 2021-2022

Something that jumps out is to see Tkachuk at the very top while Gaudreau and Lindholm are near the bottom of the team. The three played most of their seasons together and were one of the most prolific lines in 2021-2022. The expectation would be for their ratings to be similar. This suggests that Tkachuk performed very well when away from Gaudreau and Lindholm while they did not perform as well. This would cause Tkachuk to have a slightly higher rating than the other two. As the model continues to iterate through the seasons, Tkachuk gets further and further away from the other two and then gets more credit for the success of their line. This results in Tkachuk having a much higher rating

than the other two. It is interesting to note that for the 2022-2023 season, all three players would end up on different teams. Tkachuk was traded to Florida while Gaudreau signed as a free agent with Columbus. As of the time of writing this report, Tkachuk is having another great season while Lindholm and Gaudreau are failing to recreate the numbers they produced in 2021-2022. Tkachuk might well have been the main driver in the success of that line, but it is impossible to say for sure as there are many more factors impacting their play in 2022-2023.

Player	S-Rank	R-Rank	GP	Points	Cap Hit	Hart	Selke	Norris
Matthews	1	8	73	106	11.6	1	10	-
McAvoy	2	113	78	56	4.9		-	4
Bergeron	3	11	73	65	6.9		1	-
McDavid	4	1	80	123	12.5	2		-
Weegar	5	141	80	44	3.3		-	14
Marner	6	157	72	97	10.9		16	-
Bunting	7	106	79	63	1.0			-
Marchand	8	127	70	80	6.1		12	-
Pastrnak	9	197	72	77	6.7			-
M. Tkachuk	10	4	82	104	7.0	14	27	-
Grzelcyk	11	68	73	24	3.7		-	
Makar	12	183	77	86	9.0	8	-	1
Reinhart	13	110	78	82	6.5		27	-
D. Toews	14	112	66	57	4.1		-	8
M. Reilly	15	96	70	17	3.0		-	
Ekblad	16	245	61	57	7.5		-	6
J. Robertson	17	3	74	79	0.8	13		-
J. Slavin	18	7	79	42	5.3		-	9
Puljujarvi	19	155	65	36	1.2		27	-
T. Hall	20	76	81	61	6.0			-

Table 5.6: Top 20 Ranked Players Using Single 2021-2022 Season

Finally let's compare the rankings we got when running the events for the 2021-2022 season through the model once versus repeatedly running the season events through the model 100 times. Tables 5.6 and 5.7 show the single season rank (S-Rank), the rank after running the season repeatedly 100 (R-Rank), the number of games played by the player during that season, the number of points they scored over the season, their salary cap hit

Player	S-Rank	R-Rank	GP	Points	Cap Hit	Hart	Selke	Norris
McDavid	4	1	80	123	12.5	2		-
Marchment	25	2	54	47	0.8		18	-
J. Robertson	17	3	74	79	0.8	13		-
M. Tkachuk	10	4	82	104	7.0	14	27	-
Meier	112	5	77	76	6.0			-
Garland	43	6	77	52	5.0			-
J. Slavin	18	7	79	42	5.3		-	9
Matthews	1	8	73	106	11.6	1	10	-
Raffl	757	9	76	16	1.1		27	-
Milano	124	10	66	34	1.7			-
Bergeron	3	11	73	65	6.9		1	-
Fox	99	12	78	74	0.9		-	5
Doughty	33	13	39	31	11.0		-	
Asplund	160	14	80	27	0.8			-
Kadri	80	15	71	87	4.5		27	-
Brodin	37	16	73	30	6.0		-	
Kaprizov	57	17	81	108	9.0	7		-
Mikheyev	154	18	53	32	1.6			-
Perfetti	273	19	18	7	0.9			-
Petry	559	20	68	27	6.3		-	

Table 5.7: Top 20 Players Repeatedly Using 2021-2022 Season 100 Times

(or average salary) in millions of dollars during that season as well as their rank in the voting for three different individual awards. The Hart trophy is given to the player judged to be the most valuable to his team during the regular season. The Selke trophy is given to the best defensive forward. The Norris trophy is given to the best defenseman. All three awards are voted on by the Professional Hockey Writers' Association. Player that are not eligible for a particular award are marked with "-".

The top players in Table 5.6 all have played many games that season. This makes sense since they have more samples to increase their rating. The players are slightly more likely to have been voted for an award, especially for the Norris. Only 2 of the 8 defensemen in the top 20 did not receive votes for best defenseman. The winner of all 3 awards are present in the top 20. Most players have a high point total, especially the forwards. The exceptions seem to be players on very good teams that have other teammates in the top

20. Matt Grzelcyk and Mike Reilly are both defenseman for Boston which have 5 other players in the top 20: McAvoy, Bergeron, Marchand, Pastrnak and Taylor Hall. They are rated so highly due to playing on such a good team. In fact, all Boston players have a much lower ranking after 100 runs with only Bergeron staying in the top 20. This seems to be a trend for teams with multiple players in the top 20. Of the top Toronto line of Matthews, Marner and Bunting, only Matthews remains near the top of the rankings in the R-Rank. The player that falls the furthest in ranking after running the model over the season 100 times is Aaron Ekblad. Similar to the Boston and Toronto players, Ekblad could be the victim of the strength of his teammates. He played on a very deep Florida team that dominated their opponents for most of the year, so the model might not give him as much credit for his success as time went on.

In Table 5.7, we see many players who have played fewer games than the players in Table 5.6. The player with the fewest games is Cole Perfetti for Winnipeg with 18 games. The fact he is in the top 20 players with only 18 games does suggest that we may have run the events through the model too many times for the season if such a small sample size can bring someone near the top of the league. We also see less players who were nominated for an award. There are more players making less than $2 million dollars in the top 20. It is important to note that Jason Robertson, Adam Fox and Cole Perfetti are all on their entry level contract which limits how high their salary can be for the first few years in the league. Both Fox and Robertson signed contracts with a much higher salary after the 2021-2022 season. Perfetti still has two years remaining on his entry level contract. There are some lesser known names and players with smaller point totals. A few players rose very high in the rankings after 100 runs, but none more than Raffl which we had discussed earlier in the results.

Results after running the model over the season once seem to give a better indication of the total success a player has seen over the season while the results after the 100^{th} run seem to offer more insight into how players performed given the quality of their teammates and competition. While both do provide interesting information, running the model many times over the data on a season might offer the most interesting and less obvious insights. There is a balancing act on how many times the model should be ran. This is a topic that could be explored further.

Chapter 6

Conclusion

In conclusion, an event-based TrueSkill model produced promising results through simulation. The changes in player rating were much more gradual and could detect the underlying skill faster if enough events per game were present. It also succeeded in distinguishing between players within a team when the players changed during a game. Game context which could have an impact on the outcome of an event was able to be detected by the model and properly accounted for.

When applying the event-based TrueSkill model to hockey, several hurdles were discovered. When detecting player skill during certain events, like a man advantage, the model implementation fails to take into account the passage of time. There are times when the absence of an event is positive. A potential improvement on the implementation would have been to add artificial events when there was a gap in the observed events. This would require adding ties into event-based TrueSkill models.

Furthermore, the weightings on the events were such that no meaningful player ranking was able to be achieved when running all the events through the model. To reach a stable estimate of the player's underlying skill, the events were rerun through the model several times. There seem to be benefits to this method. The quality of teammates and of competition would be properly accounted for when updating player rankings based on the outcome of events. However, more investigation might need to be done to ensure this method produces the desired outcome.

Therefore, the event-based TrueSkill model does seem to have promise in modeling fluid games such as hockey. I can lead to interesting insights as shown in this report. However, more work could be done to make the implementation more robust.

References

[1] Arvind. Advanced stats 102 - what is gar? https://www.pensionplanpuppets.com/2017/6/16/15774850/advanced-stats-102-what-is-gar-dawson-sprigings-nhl, 6 2017. [Online; accessed 04-October-2022].

[2] John Doe. Composite tailored regression modeling for evaluative ratings in professional hockey. 2016.

[3] Arpad E. Elo. *The Rating of Chessplayers, Past and Present.* Arco Pub., New York, 1978.

[4] John Fischer. Blocked shots, the new jersey devils' rink, and scorer bias - a follow up. https://www.allaboutthejersey.com/2010/7/8/1559914/blocked-shots-the-new-jersey, 7 2010. [Online; accessed 03-October-2022].

[5] Mark E. Glickman. Parameter estimation in large dynamic paired comparison experiments. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 48, 1999.

[6] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill(tm): A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press, January 2007.

[7] Drew Hynes. nhlapi. https://gitlab.com/dword4/nhlapi, 2022.

[8] Julia Ibstedt, Elsa Rådahl, E Turesson, and Magdalena vande Voorde. Application and further development of trueskill™ ranking in sports. 2019.

[9] Ken Krzywicki. Shot quality model. http://hockeyanalytics.com/Research_files/Shot_Quality_Krzywicki.pdf, 1 2005. [Online; accessed 04-October-2022].

[10] Paul Laberge. nhlstats. https://github.com/plaberge/nhlstats, 2022.

[11] Gustavo Landfried. Real examples — trueskillthroughtime. https://glandfried.github.io/TrueSkillThroughTime.jl/man/examples/#The-History-of-the-Association-of-Tennis-Professionals. [Online; accessed 30-September-2022].

[12] Dom Luszczyszyn. Measuring single game productivity: An introduction to game score. https://hockey-graphs.com/2016/07/13/measuring-single-game-productivity-an-introduction-to-game-score/, 7 2016. [Online; accessed 04-October-2022].

[13] Dom Luszczyszyn. By the numbers: Can new leafs defenceman ron hainsey handle tough minutes? https://theathletic.com/75315/2017/07/13/by-the-numbers-can-new-leafs-defenceman-ron-hainsey-handle-tough-minutes/, 7 2017. [Online; accessed 04-October-2022].

[14] Tom Minka, Ryan Cleven, and Yordan Zaykov. Trueskill 2: An improved bayesian skill rating system. Technical Report MSR-TR-2018-8, Microsoft, March 2018.

[15] Alan Nevill and R Holder. Home advantage in sport: an overview of studies on the advantage of playing at home. *Sports medicine (Auckland, N.Z.)*, 28:221–36, 11 1999.

[16] Emmanuel Perry. Shot quality and expected goals: Part i. https://corsica.hockey/blog/2016/03/03/shot-quality-and-expected-goals-part-i/, 3 2016. [Online; accessed 04-October-2022].

[17] Emmanuel Perry. The art of war. https://corsica.hockey/blog/2017/05/20/the-art-of-war/, 5 2017. [Online; accessed 04-October-2022].

[18] Greg Revak. What's more important, shot quantity or shot quality? https://hockeysarsenal.substack.com/p/shot-quantity-or-shot-quality, 9 2020. [Online; accessed 04-October-2022].

[19] Alan Ryder. Shot quality. http://hockeyanalytics.com/Research_files/Shot_Quality.pdf, 1 2004. [Online; accessed 04-October-2022].

[20] Michael E. Schuckers and James Curro. Total hockey rating (thor): A comprehensive statistical rating of national hockey league forwards and defensemen based upon all on-ice events. 2013.

[21] Kevin Sidwar. The undocumented nhl stats api. https://www.kevinsidwar.com/iot/2017/7/1/the-undocumented-nhl-stats-api, 7 2017. [Online; accessed 09-November-2022].

[22] Luke Solberg. A new look at aging curves for nhl skaters (part 1). https://hockey-graphs.com/2017/03/23/a-new-look-at-aging-curves-for-nhl-skaters-part-1/, 3 2017. [Online; accessed 04-October-2022].

[23] Luke Solberg. Wins above replacement: History, philosophy, and objectives (part 2). https://hockey-graphs.com/2019/01/16/wins-above-replacement-history-philosophy-and-objectives-part-2/#more-23257, 1 2019. [Online; accessed 04-October-2022].

[24] Brandon Storey. How do shifts work in hockey? (explained for beginners). https://hockeyquestion.com/how-hockey-shifts-work/#:~:text=A%20hockey%20shift%20is%20when,give%20players%20time%20to%20rest. [Online; accessed 22-November-2022].

[25] Lu Xin, Mu Zhu, and Hugh Chipman. A continuous-time stochastic block model for basketball networks, 2015.

[26] Alexander Yu. Do stats matter in basketball? https://itsxandery.com/do-stats-matter-in-basketball/, 5 2021. [Online; accessed 04-October-2022].

APPENDICES

Appendix A

NHL Public data API

The NHL has public APIs with game data that were used to get the data used in this report. There is no official publicly available documentation for these APIs, however many people have documented many of the endpoints online. The APIs are covered by the following copyright.

"NHL and the NHL Shield are registered trademarks of the National Hockey League. NHL and NHL team marks are the property of the NHL and its teams. © NHL 2022. All Rights Reserved."

A.1 Game Data Endpoints

The main NHL API has many different end points and a lot of available data. A high level overview of the API's endpoints was done by Kevin Sidwar [21]. He links to the work of Drew Hynes [7] and Paul Laberge [10] which have more extensive documentation of the endpoints. For the work done in this report, the following APIs were used.

A.1.1 Seasons

`https://statsapi.web.nhl.com/api/v1/seasons`

Returns a JSON list of all NHL seasons with data. Each season has a "seasonId" property which is used in other endpoints as a unique identifier.

A.1.2 Teams

`https://statsapi.web.nhl.com/api/v1/teams`

Returns a JSON list of all NHL active teams. Each team has an "id" property which is used in other endpoints as a unique identifier. We can get high level information about teams from this endpoint.

A.1.3 Schedule

`https://statsapi.web.nhl.com/api/v1/schedule?season=@seasonId&gameType=R`

Returns a JSON list of all regular season NHL games for a given season. The season is specified using the season id found in A.1.1. Each team is referenced using the team id found in A.1.2. From this endpoint, we can get all the game unique ids from the parameter "gamePk". We can also determine which team is the home team.

A.1.4 Game Events

`https://statsapi.web.nhl.com/api/v1/game/@gameId/feed/live`

Returns a JSON with a list of all events for the specific game as well as the game boxscore and individual player stats. We use this endpoint primarily for the list of game events. The game is specified using the game id found in A.1.3.

A.2 Shifts Data Endpoints

Shifts data was queried from the following API:

`https://api.nhle.com/stats/rest/en/shiftcharts?cayenneExp=gameId=@gameId`

"*@gameId*" needs to be replaced with the specific game for which you want shift data. The game ids match those found in A.1. The API returns a JSON object which contains a list of shift objects as well as goals. This API was only used for the shift data. Goals were ignored. Each shift object represents a single shift from a player during the game. The shift object containts the following properties:

- id : Globally unique identifier for the shift
- detailCode: Did not use. (Gives information on goals)
- duration: Length of shift
- endTime: Exclusive end of shift (time starts at 00:00 and counts up)
- eventDescription: Did not use. (Gives information on goals)
- eventDetails: Did not use. (Gives information on goals)
- eventNumber: Did not use
- firstName: Player first name
- gameId: Global game identifier matching those found in in A.1
- hexValue: Team colour
- lastName: Player last name
- period: Numerical value of period (starting at 1. Overtime is 4)
- playerId: Global player identifier matching those found in in A.1
- shiftNumber: Shift count for player
- startTime: Inclusive start of shift (time starts at 00:00 and counts up)
- teamAbbrev: Team name abbreviation
- teamId: Global team identifier matching those found in in A.1
- teamName: Team name
- typeCode: Type of event. Shifts have typeCode = 517

A.3 Notes From Working with API Data

Events with null location ids were ignored with the exception of penalties, since location data was not used to assess the impact of the event, and takeaways since there were many missing location data. This resulted in excluding 2 blocked shots, 1 giveaway, 1 hit, 2 shots. For the 474 takeaways missing location data, they were assumed to have happened in the neutral zone.

Three goals were missing location values in the data. They were manually updated since goals were critical to many metrics and not just as individual events. The goal scored by Oshie in the third period of game 2018020993 on the 1st of March, 2019 was given (X,Y) coordinates of (45, -40). The second goal scored by Schaller in the first period of game 2018021114 on March 17th, 2019 was given (X,Y) coordinates of (67, 10). The goal scored by Burakovsky in the first period of game 2018021122 on March 19th, 2019 was given (X,Y) coordinates of (-65, -22).

Location data have (X,Y) coordinates where X represents the length of the ice and Y represents the width. The center of the ice has coordinates (0,0) and the coordinates are bounded by $|X| <= 100$ and $|Y| <= 42$. Any event have occurred in the neutral zone if $|X| <= 25$. To determine whether a coordinate is in the offensive zone, you need to know the period as teams switch sides after every period. The home team will always start on the same side in each game. However, which side that is depends on the team. The following teams will be in the offensive zone if $X > 25$ in the first or third periods and they are at home: Anaheim Ducks, Arizona Coyotes, Boston Bruins, Buffalo Sabres, Columbus Blue Jackets, Edmonton Oilers, Nashville Predators, New Jersey Devils, New York Islanders, Ottawa Senators, Pittsburgh Penguins, San Jose Sharks, Seattle Kraken, St. Louis Blues, Tampa Bay Lightning. The remaining teams would be in the defensive zone if $X > 25$ in the first or third periods and they are at home.