A Manifesto of Skill

A Journey Through Competitive Rating Systems

Jordan Stout

April 30, 2025

Abstract

This paper explores the evolution of skill estimation models, from the early days of informal rankings to modern systems. We begin with the Elo System, which revolutionized chess ratings through its simple yet effective probabilistic model. We then examine the Glicko system, which enhance Elo by introducing measures of uncertainty and time-based updates. Finally, we analyze Microsoft's TrueSkill framework, a Bayesian model capable of handling multiplayer dynamics, uneven teams, and free-for-all matches.

1 Introduction

Under the hood of every competitive game player is the underlying desire to be better than the rest. Thus, it is desirable for there to be a system which ensures a quantifiable, reliable, and fair method to reward players for their wins, and penalize them for their losses. It is also important to make sure that players are given a fair opportunity to increase and decrease their skill rating.

In 1960, chess was the first major game to introduce a formal, statistically grounded rating method which we all know as the Elo System. With the rise of online gaming, however, the limitations of the Elo system became apparent, particularly in games with large player bases, frequent matchups, and varying formats. This led to the development of more advanced rating systems, such as Glicko and TrueSkill, which improved upon Elo by accounting for factors like uncertainty, time decay, and multi-player dynamics, ensuring a more accurate reflection of player skill in fast-paced gaming environments.

2 Pre-Elo Era

Before Arpad Elo made his legendary contribution to the scene, the chess rating system was very informal. Players received their titles (e.g. Master, International Master, Grandmaster) through the opinions of their peers, tournament judges, notable wins, and other qualitative measurements.

In the early 1950's, the Harkness system was adopted by U.S. Chess Federation (USCF) in a bleak attempt to quantify players' skill. After each tournament, a player's skill rating would be adjusted based on the average rating of their opponents played. If at a tournament, you played players with an average rating of r with w wins and l losses, your rating would be updated following Equation 1.

$$r' = r + (w - l) \times 50 \tag{1}$$

3 Rise of Elo

The arrival of the Elo System in 1960 occurred during a legendary moment in the history of chess dubbed the Cold War chess era. While the Soviet Union and the United States were pointing their nukes at each other. Chess nerds from both countries were enlisting in their own strategy game-based proxy war. It was during this time Arpad Elo's system was accepted by the USCF in 1960, then by FIDE in 1970.

The systems beauty was in its simplicity. It assumes that the difference between two players' ratings follow a logistic distribution. The CDF of which can be seen in Equation 2.

$$F(x \mid \mu, s) = \frac{1}{1 + e^{-(x-\mu)/s}} \tag{2}$$

Off of this, Arpad defined the equation to find the expected outcome of a chess match, where R_A is player one's rating and R_B is player two's.

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}} \tag{3}$$

The formula to update a player's Elo score after a game can be seen in Equation 5. In the equation, R_A is the players pregame Elo score, R'_A is the players updated score, K is a constant that decides how fast scores increase and decrease in the system, and S_A take on a value following Equation 4.

$$S_A = \begin{cases} 1 & \text{if Player A wins} \\ 0.5 & \text{if the game is a draw} \\ 0 & \text{if Player A loses} \end{cases} \tag{4}$$

$$R'_A = R_A + K(S_A - E_A) \tag{5}$$

To this day, the Elo system is still used in USCF and FIDE tournaments with some minor adjustments to the K value. One interesting thing to note is that that there is a finite amount of Elo to be won or lost ignoring the 1500 added whenever a new player enters the pool.

4 The Glicko System

4.1 Glickman's Vision

Imagine Joe Chess-Player had an Elo score of 2100 in 1980 but then quit to start a family. Fast forward 30 years, he's an empty-nester and decides to make his triumphant return to the competitive chess scene. FIDE ranks him right where he left off, 2100. Should he be gaining and losing Elo score at the same rate as other active players, or should be give the system a chance to see if Joe is still in his prime?

Harvard's Dr. Mark Glickman thought the gain/loss rate should be adjusted, which is why in 1999 he released The Glicko system. An improvement upon the Elo system with the addition of a Rating Deviation (RD) score which measures uncertainty around a players rank. The higher a players RD, the more uncertain the system is about the player's skill level, and therefore the players Elo score will fluctuate until the system is confident it has found the player's true Elo.

4.2 Implementation

The Glicko system displays a players skill R_A as the mean of a Gaussian distribution where $s \sim \mathcal{N}(R_A, RD^2)$. One key feature of the Glicko system that must be understood is the concept of a rating period. The name is pretty self-explanatory but in terms of the Joe Chess-Player hypothetical, he would have ended his rating period in 1980 when he quit then began a new one 30 years later. When a player enters the system, they are given a default RD of 350.

The expected outcome of player A winning a chess match against player B under Glickman's system is modeled using Equation 6.

$$E_A = \frac{1}{1 + 10^{-g(RD_B)(R_A - R_B)/400}} \tag{6}$$

This should be very reminiscent of the expected outcome under the Elo system with the addition of the dampening function $g(RD_B)$ which accounts for the rank uncertainty in the player's opponent and is defined as Equation 7 where q is a constant.

$$g(RD_B) = \frac{1}{\sqrt{1 + \frac{3q^2 RD_B^2}{\pi^2}}} \tag{7}$$

Without going through all the details of the algebra we can understand this intuitively by noting that

- If RD_B is small (we trust the opponent's rating), then $g(RD_B) \approx 1$, and the rating update is more sensitive.
- If RD_B is large (we're unsure about the opponent), then $g(RD_B) \ll 1$, reducing the impact of the game on your rating.

The rank update equation of the Glicko system can be seen below in Equation 8. The system is summing across all players player A has played in the current rating period. S_i in this equation is the same S as Equation 4 for each opponent played.

$$R'_A = R_A + \frac{q}{\frac{1}{RD_A^2} + \frac{1}{d^2}} \sum_{i=1}^{N} g(RD_i)\,(S_i - E_i) \tag{8}$$

To break this down in a way that it within the bounds of this paper, we can first note that $g(RD_i) \in (0, 1]$ and $S_i - E_i \in [-1, 1]$ which means the range of the whole summation is $[-N, N]$. It's a scalar that represents how surprising each of the player's wins was. The quotient on the left represents a weighted average of the uncertainty in player A's rating and the system's estimated variance.

5 TrueSkill

5.1 Introduction to TrueSkill

With the rise in popularity of online videogames, the need for more robust skill rating models has risen. The Elo and Glicko systems are useful for 1v1 scenarios, but many online games require more complex interactions such as free-for-all gamemodes, team based matchmaking, and the intricate handling of draws. In 2006, a team from Microsoft Research release a solution to these issues in the form of TrueSkill.

5.2 A Bayesian Approach

TrueSkill takes a Bayesian approach to skill rating. It implements each player's skill as a Gaussian probability distribution (Equation 9). During the match, it models individual performance (Equation 10) and team performance (Equation 11) as noisy expressions of those skills.

$$s_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \tag{9}$$

$$p_i \sim \mathcal{N}(s_i, \beta^2) \tag{10}$$

$$t_j = \sum_{i}^{n} p_i \tag{11}$$

After the match, the observed outcome is used to update the players' skill distributions via approximate Bayesian inference.

$$p(\mathbf{s} \mid \mathbf{r}, A) = \frac{P(\mathbf{r} \mid \mathbf{s}, A) \cdot p(\mathbf{s})}{P(\mathbf{r} \mid A)} \tag{12}$$

Where

- $\mathbf{s}$ is a vector of all participating players' skills
- A represents team assignments during the match

- $p(\mathbf{s})$ represents the joint prior belief about the skills of all players before observing any game outcomes. It is defined by Equation 13.

$$p(\mathbf{s}) = \prod_{i=1}^{n} \mathcal{N}(s_i; \mu_i, \sigma_i^2), \tag{13}$$

- $\mathbf{r}$ is a vector that represents the results of the match
  - For example given a match with 3 teams A, B, and C and the result of the match was A won and B and C tied for second, the resulting $\mathbf{r}$ would be $\mathbf{r} = (1, 2, 2)$

To handle draws, TrueSkill implements a draw margin ϵ. For two teams' results to be declared a draw, the difference in team performance must be less than ϵ. Expressed mathematically this is to say $r_j = r_{j+1} \Rightarrow |t_j - t_{j+1}| < \epsilon$

5.3 Factor Graphs

Before describing the implementation of TrueSkill, we must first explore the topic of factor graphs. A factor graph is a graph that represents the factorization of a function. Lets consider the function $g(\mathbf{X})$ represented in Equation 14 where $S_i \subseteq \{X_1, ..., X_n\}$.

$$g(X_1, ..., X_n) = \prod_i f_i(S_i) \tag{14}$$

Breaking a function down into a product of smaller functions is a very useful tool in the context of probability distributions because a factor graphs allows us to represent a joint probability distribution into a product of probability distributions that depend on a smaller number of variables. After breaking joint probability distributions into factor graphs, we are able to run algorithms on them efficiently.

Let's visualize this using an example.

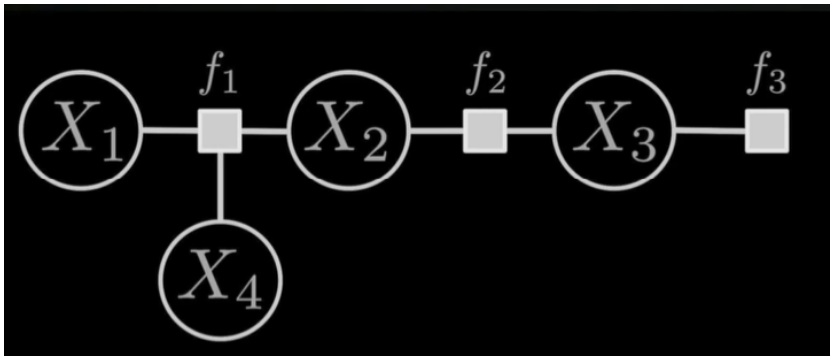

Figure 1: Example factor graph

This factor graphs represents the function $g(X_1, ..., X_4) = f_1(X_1, X_2, X_4)f_2(X_2, X_3)f_3(X_3)$. We can begin to see how this can be applied in TrueSkill when looking at TrueSkill's prior distribution in Equation 13.

5.4 TrueSkill Factor Graph Implementation

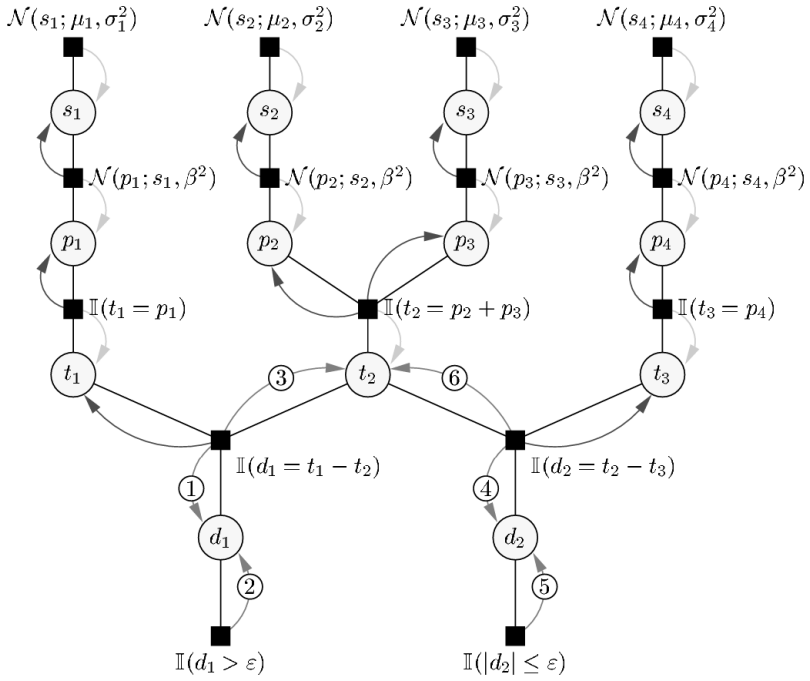

Figure 2: Factor graph for an example 3 team, 4 player scenario where $\mathbf{r}=(1,2,2)$. Player 1 is alone on team 1, players 2 and 3 are together on team 2, and player 4 is alone on team 3. Mathematically this can be represented by $A_1=\{1\}$, $A_2=\{2,3\}$, and $A_3=\{4\}$. This entire graph is a visual representations of the joint distribution $p(\mathbf{s},\mathbf{p},\mathbf{t}|\mathbf{r},A)$.

Before moving on, let's note three facets of the graph to give some context in this scenario. Firstly, moving from top to bottom we can see an individual players skill s_i is related to the individuals performance p_i, which is related to the teams performance t_j, which is related to the difference in t_j and t_{j+1} represented by d. Secondly, TrueSkill's joint prior distribution can be represented by the product of the top row of factors. Finally, the product of all other factor makes up the likelihood for the game's outcome given these players and teams.

Going back to the original goal of this system which is to approximate the posterior distribution for an individual player $p(s_i|\mathbf{r},A)$ to update their skill belief accordingly. We can extract this through Equation 15.[1]

$$p(s_i|\mathbf{r},A)=\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}p(\mathbf{s},\mathbf{p},\mathbf{t}|\mathbf{r},A)d\mathbf{p}d\mathbf{t} \tag{15}$$

To solve this double integral is you have to integrate out all the performance and team score variables, which in practice means a high-dimensional, non-Gaussian integral. The factor graph allows the system to decompose the full joint distribution into local parts and use a method called approximate message passing to update all of the values.

[1] I rationalize this process by relating it to finding the volume of a standard 3D polynomial where you first integrate over x and then y.

5.5 Approximate Message Passing

As discussed in the previous section, the goal of approximate message passing is to avoid having to perform the integral in Equation 15. To do this, we update the marginal distributions individually[2] with 3 equations. In these three equations, $\mathcal{F}_{v_k}$ is the set of all factors connected to variable v_k, $m_{f \to v_k}$ is the message from a function to a variable, and $m_{v_k \to f}$ is the message from a variable to a function.

1. The definition that a marginal distribution about a variable v_k is computed by multiplying all incoming messages from the factor nodes it's connected to.

$$p(v_k) = \prod_{f \in \mathcal{F}_{v_k}} m_{f \to v_k}(v_k)$$

2. Message from a factor to a variable, we will dig deeper into this later as this is where the approximation happens and is what makes TrueSkill work.

$$m_{f \to v_j}(v_j) = \int f(v) \prod_{i \neq j} m_{v_i \to f}(v_i)\, dv_{\neg j}$$

3. A variable sends a message to a factor by multiplying all incoming messages from other factors excluding the one it's sending to.

$$m_{v_k \to f}(v_k) = \prod_{\tilde{f} \in \mathcal{F}_{v_k} \setminus \{f\}} m_{\tilde{f} \to v_k}(v_k)$$

To make sense of this we will simulate the first step of the process of updating the factor graph example from Figure 2 by computing a message from factor node $f = \mathbb{I}(d_1 > \epsilon)$ to the variable node d_1 (i.e. we are updating d_1).

1. First we note that prior to the outcome we know

$$d_1 \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

$$f(d_1) = \mathbb{I}(d_1 > \epsilon)$$

.

2. Next we create a truncated Gaussian using f:

$$\hat{p}(d_1) \propto \mathcal{N}(d_1; \mu, \sigma^2) \cdot \mathbb{I}(d_1 > \epsilon)$$

3. Next we use moment matching to approximate the truncated Gaussian to a regular Gaussian. Moment matching is where you calculate the mean and variance of one distribution (the truncated Gaussian) and simply use those statistics to form a regular Gaussian. We will call this mean and variance $\mu^{'}$ and $\sigma^{'}$. Thus:

$$\hat{p}(d_1) \approx \mathcal{N}(d_1; \mu, \sigma^2) \cdot \mathcal{N}(\mu^{'}, \sigma^{'2})$$

You can see that we have now updated d_1 while taking into account $\mathbb{I}(d_1 > \epsilon)$. After this, the algorithm will perform this for every node in the factor graph which will eventually propagate down to $\mathcal{N}(s_i; \mu_i, \sigma_i^2)$ thus updating our belief of the skill rating of each player while taking all other factors into account.

[2] I rationalize this by comparing it to instead of calculating an integral numerically, you can approximate it by splitting it up into small rectangles and summing the area of them.

6 Conclusions and Further Work

As the world becomes increasingly interconnected, the demand for quality methods for competitive players to prove their superiority above others has increased exponentially. Through methods like Elo and Glicko, players have been historically ranked based on match outcomes in a statistically meaningful way. In the modern competitive video game community, even a minute improvement to such methods can enhance the experience of tens of millions of players daily. TrueSkill builds on the foundation of these earlier systems by modeling skill as a distribution rather than a single point estimate and by incorporating uncertainty, team dynamics, and match outcomes in a more nuanced way. In 2018, Microsoft Research published "TrueSkill 2: An improved Bayesian skill rating system" which introduces further refinements such as dynamic factor modeling and skill evolution over time, and it's something I'd like to look into further.

References

1. R. Herbrich, T. Minka, and T. Graepel. *TrueSkill: A Bayesian Skill Rating System.* In *Advances in Neural Information Processing Systems (NeurIPS)*, Microsoft Research, 2006.

2. T. Minka, Ryan Clevin, and Yordan Zaykov. *TrueSkill 2: An improved Bayesian skill rating system.* Microsoft Research, Technical Report, 2018.

3. M. E. Glickman. *Example of a Rating System with Uncertainty.* Boston University, 1999.

4. M. Bober-Irizar, N. Dua, and M. McGuinness. *Skill Issues: An Analysis of CS:GO Skill Rating Systems.* arXiv preprint arXiv:2410.02831, 2024.

5. A. E. Elo. *The Rating of Chessplayers, Past and Present.* Arco Publishing, New York, 1978.

Appendix

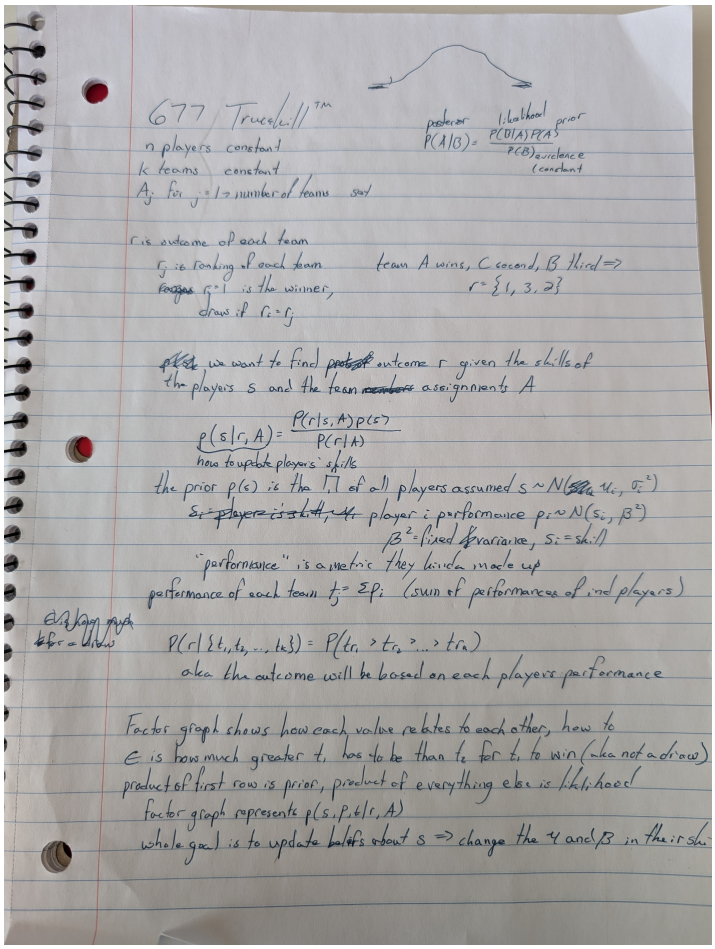

Figure 3: Notes I took while reading