EFFECT OF TEAM SIZE ON SKILL RATING CONVERGENCE

BY SEAN G MEYER

INTRODUCTION

The purpose of this simulation is to test how varying team sizes affects the number of games needed for a player to be correctly rated by skill based, competitive, rating systems (e.g. Elo, Glicko, TrueSkill). These systems attempt to quantify the skill level of a player in a multi-player competitive environment, so that players can be matched against opponents of similar skill (to create the most balanced games possible). While it's generally accepted that such systems work quite well in one versus one scenarios (such as Chess or Go), it's less clear how larger team sizes affect these systems. This simulation is an attempt to clarify that question.

SIMULATION SETUP

The simulation was performed using the Microsoft TrueSkill[i] algorithm, specifically an open source library implementation by Jeff Moser[ii] in the C# programming language. Because of this library, which seemed to be well made and easy to use, I chose to use the C# language to do my simulation. Using the library's implementation of TrueSkill, I generated player objects with two different skill ratings: private and public. The public skill rating is the one that is the players TrueSkill rating within the population, which moves after each game played, and the private skill rating is their TrueSkill rating that represents their "actual" skill. These actual skills were used to determine which team would win any given matchup, to simulate real players of differing skills playing each other.

I generated 5000 of these simulated players, assigning each a private skill rating by randomly sampling a normal distribution (with a mean of 1700). This gave a population of players whose skill was normally distributed, with many being average and a small number being very good or very poor players. For these players I matched their public and private skill ratings, simulating that these players had all been playing enough to converge to their real skill rating. This is the population of "players" that I used for running simulations.

RUNNING THE SIMULATION

Once the population of players was set up, I simulated a player joining into this game by generating one more player, this time with a brand new public skill rating. This person had no information stored by the TrueSkill system and would be treated as a brand new player. I then assigned a private skill rating to simulate this player at different levels of skill, specifically very good or very average for this simulation. The goal was to see how many games it took until the public skill rating converged to the same rating as the private skill rating, simulating a player joining a game for the first time and the ranking system having to find their correct rating.

That player was then matched up against other players, using the TrueSkill algorithm to try and give the most even matches possible (closest to 50 percent win chance). This used the public skill rating so the matches were not actually even at first, but they would become more and more even as time went on (as the skill rating converged). This is how matching is actually done in modern games, so this simulation was done the same way. If the team size being tested was not one versus one, I matched the player with random teammates before finding opponents (once again creating

the most even matches possible). Which team won was determined by using TrueSkill to get a percent win chance for each team and then generating a random number to check if they won.

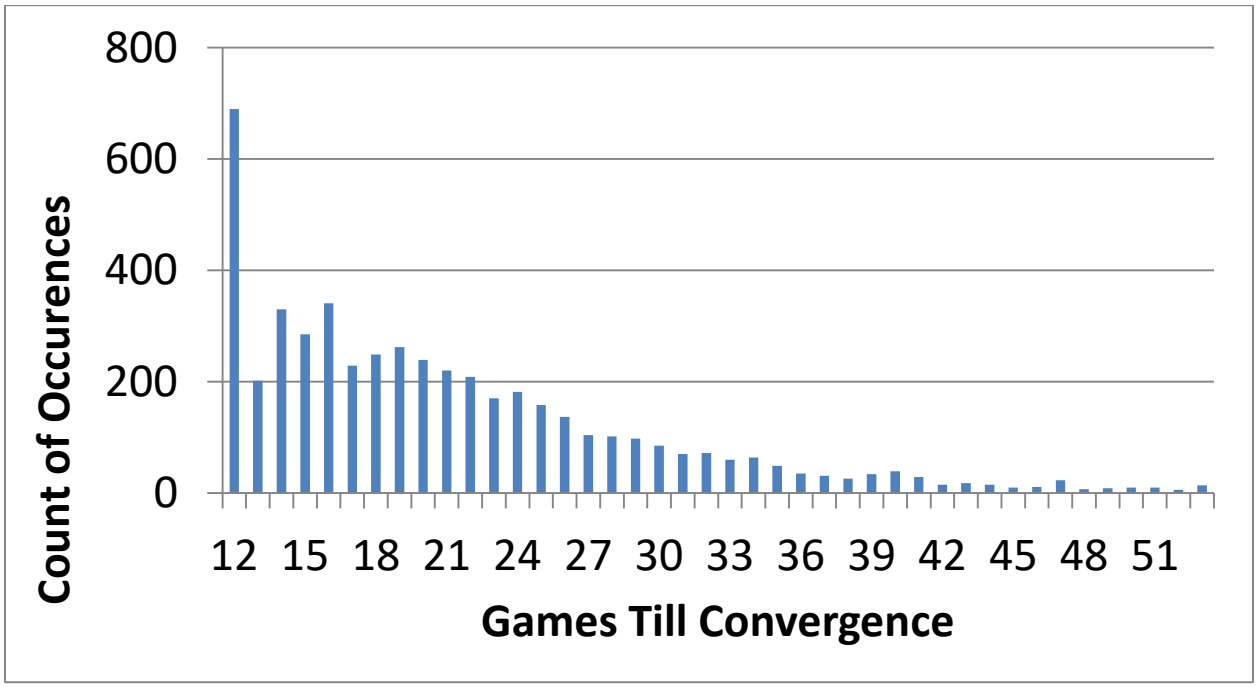
I then tracked the difference between the public and private skill ratings after each game played, and recorded how many games it took until they converged (I chose, somewhat arbitrarily, to use when the ratings were within 10 percent of each other as convergence). This was done for 1v1, 5v5, and 10v10 team sizes, and each was ran thousands of times in a monte carlo simulation to get more accurate results.

RESULTS

HIGH RATED PLAYER

First the results for a very high rated player, specifically 3 standard deviations away from the mean (top 0.3 percent of the population). Since the mean for this simulation is 1700, that puts this players skill rating at 3400.

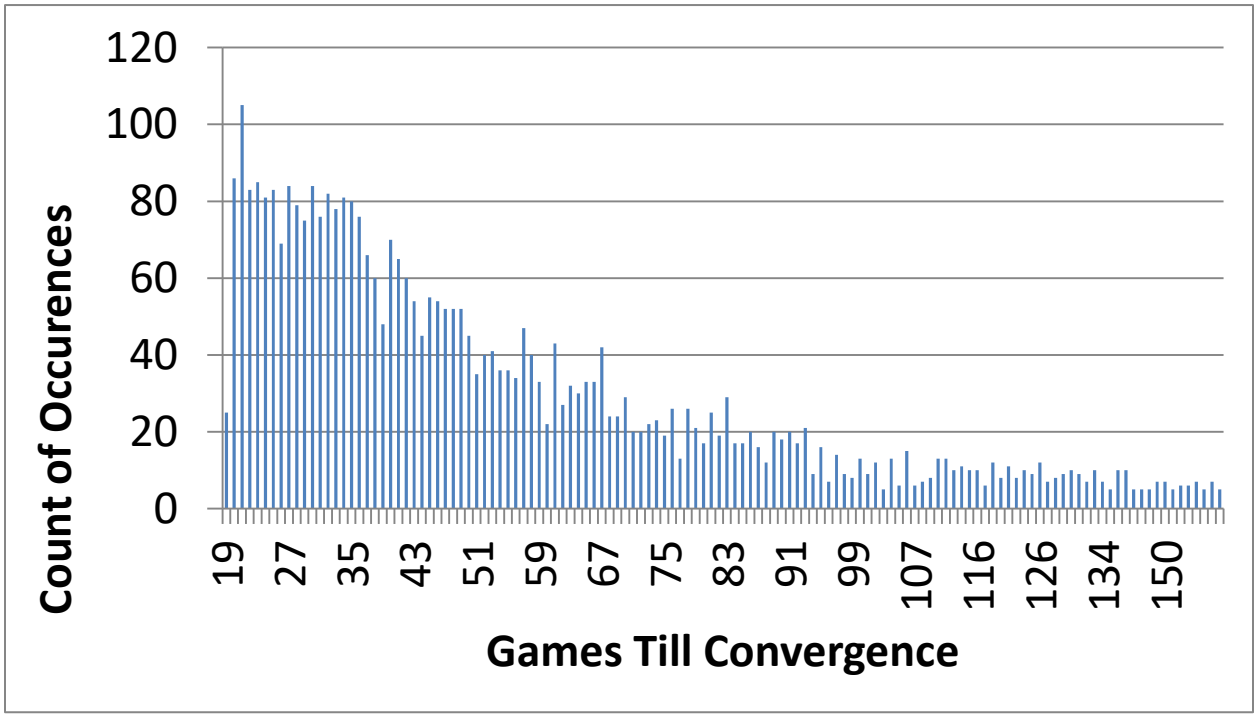
ONE VERSUS ONE



Mean – 21 Games
Mode – 12 Games

Min – 12 Games
Max – 53 Games

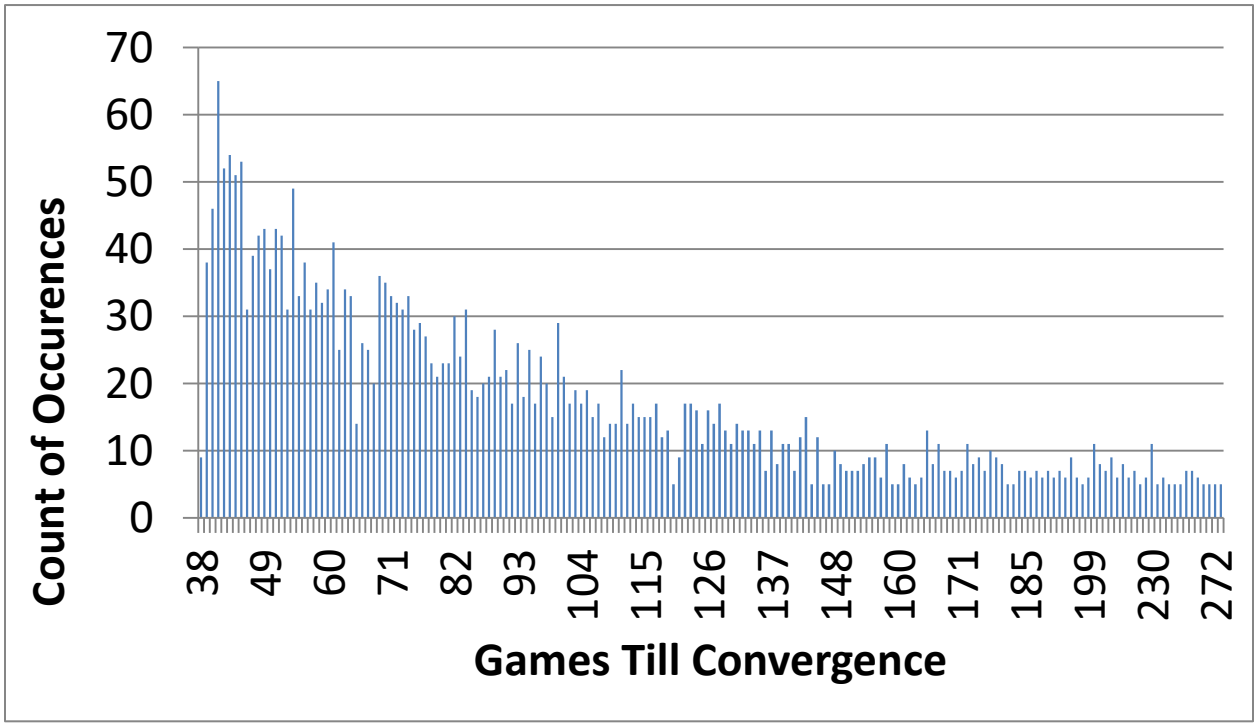
FIVE VERSUS FIVE



Mean – 64 Games
Mode – 21 Games
Min – 19 Games
Max – >300 Games

The result of max being over 300 is because only 300 games were run before individual simulations were ended. This was to lower the amount of time taken to run the simulations, which could already be quite high. This same anomaly will be seen in other results, but is only commented on here.

TEN VERSUS TEN



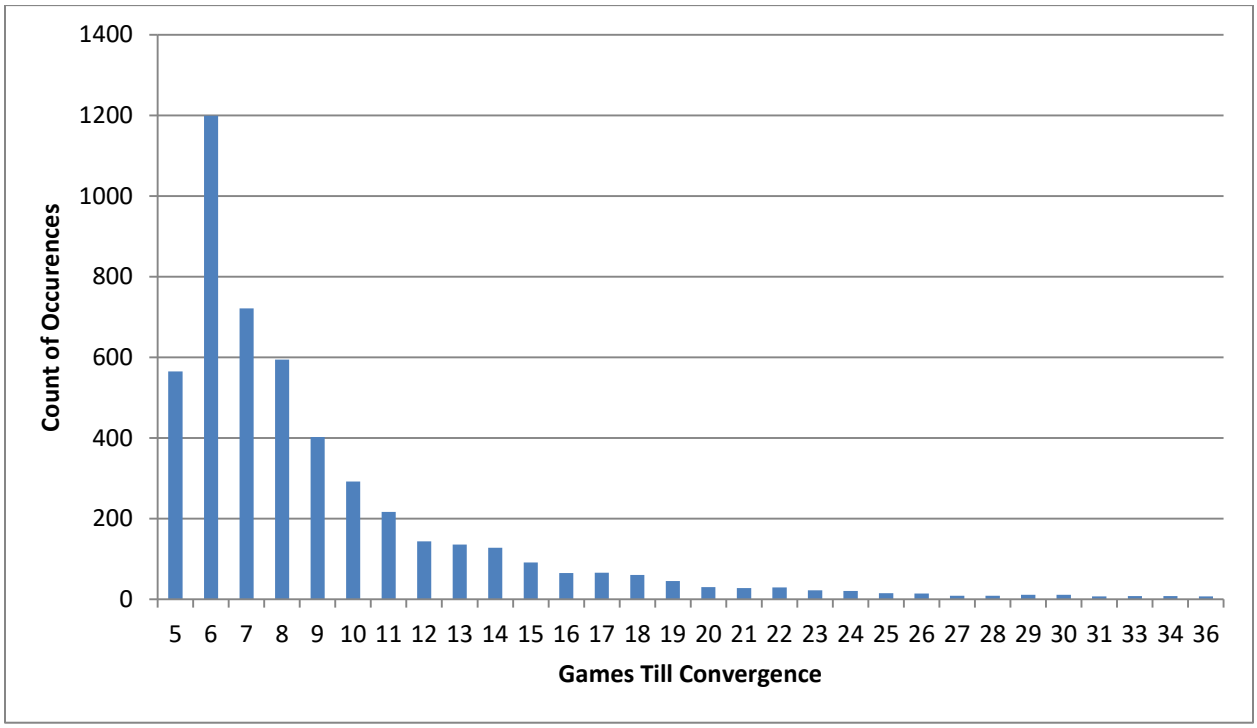
Mean – 173+ Games
Mode – >300 Games

Min – 38 Games
Max – >300 Games

AVERAGE RATED PLAYER

What follows are the results for an average rated player, specifically a player exactly at the mean of the normal distribution (in this case that is 1700).

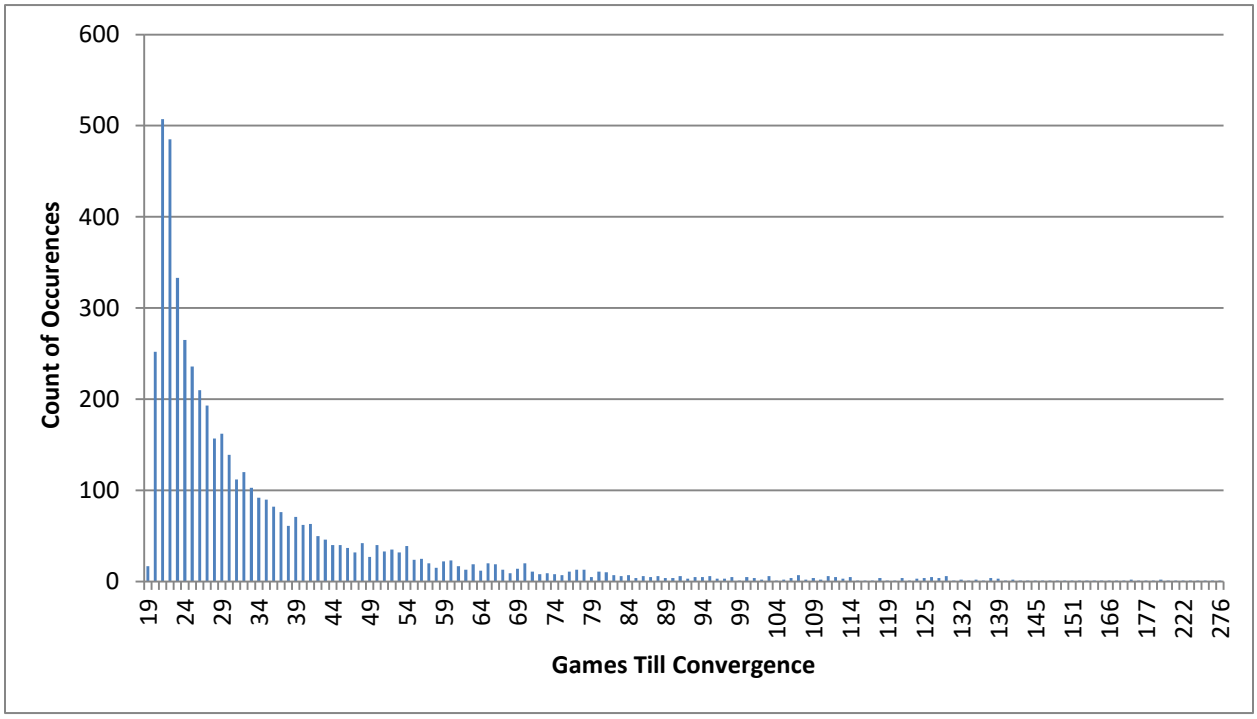
ONE VERSUS ONE



Mean – 9 Games
Mode – 6 Games

Min – 5 Games
Max – 36 Games

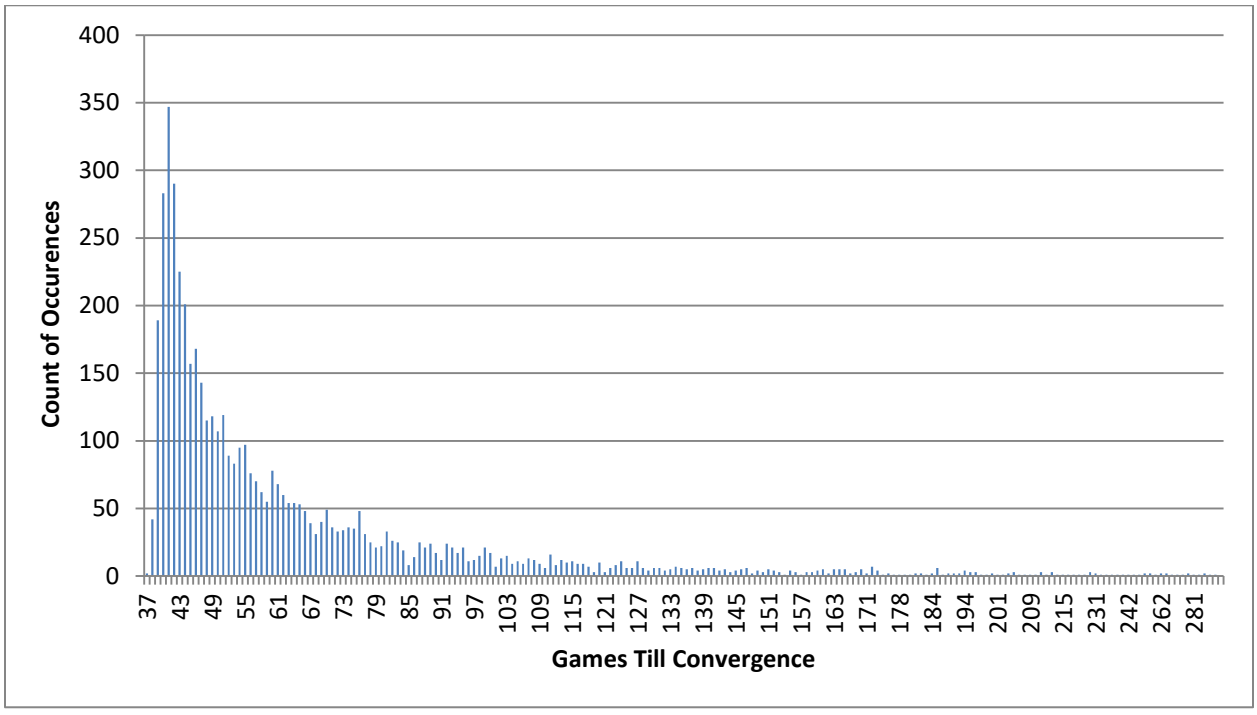
FIVE VERSUS FIVE



Mean – 35 Games
Mode – 21 Games

Min – 19 Games
Max – 266 Games

TEN VERSUS TEN



Mean – 173+ Games
Mode – >300 Games

Min – 38 Games
Max – >300 Games

CONCLUSION

Looking at how long it takes for a player's skill rating to converge as team size differs, this simulation shows that team size has a significant effect. For both an average player and a very high rated player, the larger team size moves the average up by about 3x (for both the jump to 5v5 and then the jump to 10v10). The change is less than a naive guess of multiplying the 1v1 average by number of players would give (5x for 5v5, 10x for 10v10), but the amount of games till convergence does go up quite a lot.

It's also clear that a player's rating makes a large difference on how much team size affects things. The monte carlo simulations produced results that look very similar between 1v1, 5v5, and 10v10 when looking at an average rated player. While the minimum, mean, and maximum number of games goes up as team size goes up, the overall distribution is about the same, with a very small number of outliers. By contrast the outcomes for the high rated player vary much more as team size increases to 5 and then 10. The mean goes up more than for the average player, but even more significantly the distribution of number of games taken is not the same at all. There are far more outliers, and the amount of variance is much higher, showing that extremely high rated players may take an abnormally large number of games to be correctly rated as team sizes increase.

Overall my takeaway is that team games which intend to implement a matchmaking system (using a skill rating system similar to TrueSkill) need to take extra precaution at the outliers, perhaps creating different systems just for these outliers. Similarly extremely skilled (or very bad) players of these games should be aware that their ratings, and consequently game quality, may not be correct until they have played a significant amount of games.

---

[i] TrueSkill: A Bayesian Skill Rating System. http://research.microsoft.com/pubs/67956/NIPS2006_0688.pdf
[ii] A detailed implementation of the TrueSkill algorithm. https://github.com/moserware/Skills