**An analysis of the Codeforces rating system**

Guang Yang

**<Abstract> (TBD)** (which is like a paragraph explaining the paper, just look at a few and I can help if needed)

1 research problem and objectives

2 methods

3 key results or arguments

4 conclusion

**1 Introduction**

Multi-competitor ranking is a present and on-going research area especially given the advent of massive online gaming. “They depend on a skill rating system to infer accurate player skills from historical data” (Minka et al., 2020) in order to match players with opponents similar to their levels. Microsoft Research proposed the TrueSkill method and TrueSkill 2, with TrueSkill 2 being their multi-competitor ranking method.

The summary section of a related paper, “TrueSkill 2: An improved Bayesian skill rating system” (Minka et al., 2020), says, “this paper has presented TrueSkill2, a collection of model changes to TrueSkill as well as a new system for estimating model parameters. TrueSkill2 gives significantly more accurate skill ratings than TrueSkill, measured along a variety of axes important to a game studio.” (Minka et al., 2020).

The paper begins by illustrating a set of top priority of qualities needed by a modern game studio, then continues with what TrueSkill model has satisfied and what has not. Following this is the detail of the classic TrueSkill model. TrueSkill 2 is modified in certain ways to meet the requirement omitted by the classic TrueSkill model. The various requirements, or in other words, assumptions of the model, is vital for the theory of the paper to hold. Besides, rigorous explanations on the validity of assumptions are vital as well. The explanations are usually ignored in my paper.

For the parameter estimation section, the author assigns proper values to different parameters. One purpose is to reduce ambiguity, such as fixing β to 1. The other purposes are explained in the paper to fit the design of the game itself. For different game applications, the parameters tend to differ, so game developers should adjust the model to suit better to their games.

The paper also includes the classification of confounding variables. The essential and basic part of the model is developed by disregarding those confounding variables. Then the paper classifies those variables to four categories, with elaboration in section 6, 7, 8 and 9. In each category, the author tests the algorithm, finds the problem or shortcomings, the improve the model to yield a more accurate estimation. Some features that are not added is explained in section 10, and the main reason of not adding them is that they are overlapping with previous four categories.

There is also a version of Elo for multi-competitor games. One such version is the Elo-MMR rating system, elaborated by “An Elo-like System for Massive Multiplayer Competitions” (Ebtekar & Liu, 2021). The base case is the Bayesian model for multiple competitors, similar to the first paper but more complicated with more variables. Then the author proposes the two-phase algorithm for skill estimation in detail, and the elaboration has many advanced formulas and mathematical terms. After that is the discussion on skill evolution over time, and a term “pseudodiffusion” is put forward. A set of pseudocode helps illustrate the idea. Then the paper evaluates the theoretical effectiveness of the algorithm, with calculations of time complexity and optimizations. Finally, data from past contests of different competitive programming sites such as Codeforces and TopCoder is put into the algorithm to determine the effectiveness of prediction. In the appendix part, there is also proof of theorems used in the paper.

Another version is a multi-competitor Elo method applied on Formula One matches. The article “Who’s The Best Formula One Driver Of All Time?” (Justin Moore, 2018), describes this rating method adjusted to rate the competitions with multiple competitors.

Similar to the Elo rating method, competitors are assigned an initial rating of 1300. The largest difference is that “Each session or race is treated as if it were a round-robin 1-on-1 tournament. A driver who finishes second out of 15 cars is viewed as having gone 13-1 in this tournament, losing to the first-place finisher and defeating the rest.” (Justin Moore, 2018). In this version, only competitors’ ranking will determine its rating change, but the actual scores are not taken into account; however, my intended research topic includes the effect of actual scores on rating change, so I could probably only learn the

Idea of it. Maybe I could change the simple win-lose score into a weighted version of competitor’s points.

The article also points out that artificial adjustment on rating changes is necessary, in order to prevent rating inflation or deflation. Without the adjustment, the initial uniform standard for determining competitor’s ability would fluctuate over time, certainly unfair for different competitors that stay active in different time.

Other variations also exist such as the Massey method. In thesis “Ranking Methods for Olympic Sports: A Case Study by the U.S. Olympic Committee and the College of Charleston” (GREENE et al., 2014) uses several ranking methods to evaluate the strength of US Men’s Ice Hockey team. It concludes with the global ranking of the team, the chance to win medal in the 2014 Olympics, and the improvement of the team.

For the head-to-head sports, the paper uses the Massey method, the Elo method, and the TrueSkill method to analyze the US Men’s Ice Hockey team’s placement over time. After this is the comparison of the methods. In the comparison part, the Elo rating method is classified as straight Elo (holding k value constant), simple weighted and heavy weighted. This is a signal that I may classify the Elo rating method in my paper in those three cases. Straight Elo rating method predicts better result than the other two variations of the Elo method. Then the passage evaluates the predictions of the three rating methods quantitively over time.

The major content of this paper is head-to-head sports. Rating systems on multi-competitor sports are mentioned but unfortunately, they cannot be analyzed in the same way as head-to-head sports.

In this paper, I will design and analyze the rating system for Codeforces, the most famous website that hosts international competitive programming competitions. The effectiveness of the rating system will be evaluated based on data from past contests.

**2 Problem description**

Codeforces, <https://codeforces.com/>, is a website that hosts [competitive programming](https://en.wikipedia.org/wiki/Competitive_programming) contests. “As of 2018, it has over 600,000 registered users” (Wikipedia, <https://en.wikipedia.org/wiki/Codeforces>), and the number of users is increasing at a progressive rate. If competitors participate in rated contests, their rating will change. I will research the way of rating change after the contest based on competitor’s performance.

**3. Description of Codeforces contest mechanism**

This part is particularly useful for readers desired to investigate the Codeforces contest mechanism.

A newly registered user has default rating 1500. There are four divisions in Codeforces contests. Div.1 requires a rating greater than 1900, Div.2 requires a rating less than 2100, Div.3 requires a rating less than 1600, and Div.4 requires a rating less than 1400. Although Div.4 existed in the Codeforces history, it was only held once in [Codeforces Round 640 (Div. 4)](https://codeforces.com/contest/1561/standings). Contests in other three divisions are held regularly.

There are two set of data for a problem, one is pretest and the other one is system test. Contestants submit their code to see if the code passes every test point in the pretest. If it was successful, then the contestant can wait until the system test that is tested after the contest ends. Only codes that passes system test earns the scores for a problem.

For the problems in a contest, every problem has an initial score, with the convention of the easiest task, A, worth of 500 points and the hardest task, E or F or G, worth of 3500 points. Other problems have different points but the point increases with difficulty. Also, there is a mechanism of problems devaluating with time. For a regular 2-hour contest, the value of a problem decreases at a rate of per minute. If the problem is successfully accomplished, then it will stop devaluating. There is penalty for submitting the problems as well. For each unsuccessful attempt that fails the pretest, the contestant loses 50 marks to the problem.

Another interesting concept, also the unique and symbolistic feature of Codeforces contests, is “hacking”. Dozens of contestants are allocated to the same room, and they can view each other’s code after successfully passed the pretest of a problem. Then contestants in the room can meticulously design some data to kill other people’s code. If it was successful, i.e., other people’s code fails the special data, then it is a “successful hack” and this set of data will be used at the system test. A successful hack brings extra 100 points to this problem; however, an “unsuccessful hack” result in a 50 points reduction. Although hacking other’s codes is fun, the risk is noticeable, and most hackings aren’t easy since the pretest usually consists of tens or hundreds of data points.

Despite all those rules, no matter how much attempts or unsuccessful hack the contestant did, when he or she solved the problem, the score for it cannot drop below 30% of its original points.

“For example, if the problem B was solved after 10 minutes of contest, then it costs points. For each attempt there is penalty of 50 points. So, if the problem B was solved after 10 minutes from the beginning with the third attempt, the score for it is points.” (<https://codeforces.com/blog/entry/456>) Table 1 is copied from official blog to help illustrate the rule.

Table 1. Codeforces contest mechanism sample

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Problem | Max. score | Min. score (30%) | Loss (points per minute) | Score at the contest end |
| A | 500 | 150 | 2 points | 260 |
| B | 1000 | 300 | 4 points | 520 |
| C | 1500 | 450 | 6 points | 780 |
| D | 2000 | 600 | 8 points | 1040 |
| E | 2500 | 750 | 10 points | 1300 |

**4 My rating system – multi-competitor Elo method**

“The Elo system was originally invented as an improved chess-rating system over the previously used Harkness system.” (https://en.wikipedia.org/wiki/Elo\_rating\_system) I decided to change this classical rating method a little to produce a multi-competitor variant.

First, I calculate the expected win rate between any two competitors.

Assume the rating for player A before the contest is and the rating for player B before the contest is , and denotes the expected probability that player A outcompetes player B, then according to the classical Elo model, we have

and

With the idea of “each session or race is treated as if it were a round-robin 1-on-1 tournament” (Justin Moore, 2018), I believe every competitor in this competition should play a round-robin as well. Therefore, I assume that every two competitors had a competition based on their rating, and the expected rank of competitor is the sum of all expected probability of player loses the match against every player. Hence,

Because is the expected rate of winning over , denotes the expected rate of losing to . However, there is a bit of imperfection here. is always because the player has equal ability when competing with oneself. In addition, the expected ranking for the best player is , which should be in conventions. So, I add to this ranking estimator.

Then compare the expected ranking with the actual ranking after the contest, label it of participant . Then the change in rating of participant should be according to the classical Elo method, in which is a factor (one idea that wasn’t implemented yet: the K value for pro players and normal players could be different). Hence, the rating of participant I after the contest should be:

The rating numbers in Codeforces are usually integers; hence, I use the function to convert the new rating to an integer.

**5 Apply it on data and check result**

The data I use here mainly comes from the Codeforces website. All the ranking and scoring for each competitor in every competition is accessible in the website. According to the Codeforces competition rules, only users with a rating greater than 1900 are eligible to participate the Div.1 contests, which is the hardest among all divisions and the type with least participants, only around 1000. This data size is large enough to analyze the rating system but not excessively large to waste a long time in program; therefore, I decide to collect the data of contestants’ rankings, handles (means ID in Codeforces), official old ratings and new ratings on the closest 20 Div.1 matches.

Table 2. The Closest 20 Div.1 matches

|  |  |  |
| --- | --- | --- |
| Index | Contest Name | Contest ID |
| 1 | [Codeforces Round #673 (Div. 1)](https://codeforces.com/contest/1416) | 1416 |
| 2 | [Codeforces Round #680 (Div. 1, based on Moscow Team Olympiad)](https://codeforces.com/contest/1444) | 1444 |
| 3 | [Codeforces Round #681 (Div. 1, based on VK Cup 2019-2020 - Final)](https://codeforces.com/contest/1442) | 1442 |
| 4 | [Codeforces Round #683 (Div. 1, by Meet IT)](https://codeforces.com/contest/1446) | 1446 |
| 5 | [Codeforces Round #684 (Div. 1)](https://codeforces.com/contest/1439) | 1439 |
| 6 | [Codeforces Round #687 (Div. 1, based on Technocup 2021 Elimination Round 2)](https://codeforces.com/contest/1456) | 1456 |
| 7 | [Codeforces Round #691 (Div. 1)](https://codeforces.com/contest/1458) | 1458 |
| 8 | [Codeforces Round #692 (Div. 1, based on Technocup 2021 Elimination Round 3)](https://codeforces.com/contest/1464) | 1464 |
| 9 | [Codeforces Round #694 (Div. 1)](https://codeforces.com/contest/1470) | 1470 |
| 10 | [Codeforces Round #698 (Div. 1)](https://codeforces.com/contest/1477) | 1477 |
| 11 | [Codeforces Round #700 (Div. 1)](https://codeforces.com/contest/1479) | 1479 |
| 12 | [Codeforces Round #706 (Div. 1)](https://codeforces.com/contest/1495) | 1495 |
| 13 | [Codeforces Round #707 (Div. 1, based on Moscow Open Olympiad in Informatics)](https://codeforces.com/contest/1500) | 1500 |
| 14 | [Codeforces Round #709 (Div. 1, based on Technocup 2021 Final Round)](https://codeforces.com/contest/1483) | 1483 |
| 15 | [Codeforces Round #712 (Div. 1)](https://codeforces.com/contest/1503) | 1503 |
| 16 | [Codeforces Round #715 (Div. 1)](https://codeforces.com/contest/1508) | 1508 |
| 17 | [Codeforces Round #722 (Div. 1)](https://codeforces.com/contest/1528) | 1528 |
| 18 | [Codeforces Round #728 (Div. 1)](https://codeforces.com/contest/1540) | 1540 |
| 19 | [Codeforces Round #732 (Div. 1)](https://codeforces.com/contest/1545) | 1545 |
| 20 | [Codeforces Round #736 (Div. 1)](https://codeforces.com/contest/1548) | 1548 |

With the Python code from “<https://github.com/QAQrz/Codeforces-Rating-System/blob/master/spider_txt.py>” (*Codeforces-Rating-System/spider\_txt.py at MASTER · qaqrz/codeforces-rating-system,* 2017), I scrap the data from all the 20 contests in Table 1 to get the result of the closest 20 Div.1 matches.

The predictability index, which is the variance of the difference between my predicted rating change and the actual rating change, is used to evaluate the effectiveness of this simulation. For example, when , the variance of difference between my predicted rating change and actual rating change for each match is shown in Table 2 below.

Table 3. Variance for each contest given K=16

|  |  |
| --- | --- |
| Index | Variance |
| 1 | 12351800 |
| 2 | 12264400 |
| 3 | 12522100 |
| 4 | 9933230 |
| 5 | 11726300 |
| 6 | 11629900 |
| 7 | 6985240 |
| 8 | 7865930 |
| 9 | 13459200 |
| 10 | 8258540 |
| 11 | 16091400 |
| 12 | 21632700 |
| 13 | 7489980 |
| 14 | 10703000 |
| 15 | 14283500 |
| 16 | 11994600 |
| 17 | 10279800 |
| 18 | 7041800 |
| 19 | 16533900 |
| 20 | 14036900 |

In this example, the average variance is 11854208.25, which is quite large.

Because the lower the average variance between matches, the more accurate the method is, I began changing the value and see the change of average variance.

First, with an increment of 1, the data is shown below at Table 3.

Table 4. Average Variance from different K values with increment 1

|  |  |
| --- | --- |
| K value | Average Variance |
| -5 | 988131 |
| -4 | 612885 |
| -3 | 326919 |
| -2 | 130212 |
| -1 | 22778.3 |
| 0 | 4608.41 |
| 1 | 75706.7 |
| 2 | 236068 |
| 3 | 485703 |
| 4 | 824597 |
| 5 | 1252770 |
| 6 | 1770190 |
| 7 | 2376900 |
| 8 | 3072880 |
| 9 | 3858110 |
| 10 | 4732590 |
| 11 | 5696350 |
| 12 | 6749400 |
| 13 | 7891740 |
| 14 | 9123270 |
| 15 | 10444100 |
| 16 | 11854200 |
| 17 | 13353600 |
| 18 | 14942200 |
| 19 | 16620100 |
| 20 | 18387300 |
| 21 | 20243700 |
| 22 | 22189400 |
| 23 | 24224400 |
| 24 | 26348600 |
| 25 | 28562100 |
| 26 | 30864900 |
| 27 | 33257000 |
| 28 | 35738300 |
| 29 | 38308800 |
| 30 | 40968700 |
| 31 | 43717700 |
| 32 | 46556100 |
| 33 | 49483800 |
| 34 | 52500700 |
| 35 | 55606900 |

Figure 1. Average Variance from different K values with increment 1

The minimum point of lies between -1 and 1. Then I repeat the process again but with increment of 0.1 in region to find the desirable .

Table 5. Average Variance from different K values with increment 0.1

|  |  |
| --- | --- |
| K value | Average Variance |
| -1.00 | 22778.30 |
| -0.90 | 16943.00 |
| -0.80 | 12002.90 |
| -0.70 | 7954.41 |
| -0.60 | 4798.13 |
| -0.50 | 2534.94 |
| -0.40 | 1164.01 |
| -0.30 | 686.37 |
| -0.20 | 1100.90 |
| -0.10 | 2408.68 |
| 0.00 | 4608.41 |
| 0.10 | 7700.75 |
| 0.20 | 11687.10 |
| 0.30 | 16564.10 |
| 0.40 | 22333.80 |
| 0.50 | 28998.80 |
| 0.60 | 36555.30 |
| 0.70 | 45004.30 |
| 0.80 | 54346.00 |
| 0.90 | 64577.60 |
| 1.00 | 75706.70 |

Figure 2. Average Variance from different K values with increment 0.1

According to the graph, the value for the minimum variance over all competitions is , and the average variance is 686.37.

**5 Other features (TBD)**

My program also provides another feature, which is allowing people see roughly how high the rank is required to keep their rating from falling. With a positive K value (which is always), the participant can collect all contestants with a score (in order to be counted in the standing table).

**6 A clever strategy of attaining higher scores and ratings in Codeforces (TBD)**

This part will be finished after I developed and verified my Codeforces rating algorithm. It involves the efficient allocation of competition time, and which problems (of higher value or with lower time cost) to pick up when time is really limited.

**7 Conclusions** Sometimes just a paragraph or two and any future work, if you want.

Repeat the core of my algorithm.

Show the best R-square value in my experiments to prove that the algorithm is reliable.

Briefly introduce the strategy of attaining a higher rating.

**Bibliography**

1. Minka, T., Cleven, R. & Zaykov, Y., 2020. TrueSkill 2: An Improved BAYESIAN skill rating system. *Microsoft Research*. Available at: https://www.microsoft.com/en-us/research/publication/trueskill-2-improved-bayesian-skill-rating-system/ [Accessed August 17, 2021].

2. Justin Moore, R.D.and N.P., 2018. Who's the best Formula one driver of all time? *FiveThirtyEight*. Available at: https://fivethirtyeight.com/features/formula-one-racing/ [Accessed August 17, 2021].

3. GREENE, P.E.T.E.R. et al., 2014. *RANKING METHODS FOR OLYMPIC SPORTS: A CASE STUDY BY THE U.S. OLYMPIC COMMITTEE AND THE COLLEGE OF CHARLESTON*. Available at: https://blogs.cofc.edu/math/files/2014/10/P140617006-2iy2vvx.pdf [Accessed August 17, 2021].

4. Ebtekar, A. & Liu, P., 2021. Aram Ebtekar Vancouver, BC, Canada - arxiv. *arxiv.org e-Print archive*. Available at: https://arxiv.org/pdf/2101.00400 [Accessed August 17, 2021].

5 <https://en.wikipedia.org/wiki/Codeforces>

6 https://en.wikipedia.org/wiki/Elo\_rating\_system

7 <http://codeforces.com/blog/entry/20762>

8 <https://dreamer.blue/blog/post/2018/02/26/codeforces_rating_system_algorithm.dream>

9 <https://github.com/QAQrz/Codeforces-Rating-System>

Appendix

And other useful links:­­­­­

citation guide: <https://www.mendeley.com/guides/harvard-citation-guide>

Algorithm name: multi-competitor Elo

Or Elo-MMR: check 2101.00400.pdf

Codeforce API:

<https://codeforces.com/apiHelp/methods>

<https://iq.opengenus.org/exploring-codeforces-api/>

[http://opisthokonta.net/?p=1387](http://opisthokonta.net/?p=1387：R-square)