**Project Proposal Draft**

Guang Yang

**Research question: An analysis of the Codeforces rating system**

Abstract: (optional, add later)

1 research problem and objectives

2 methods

3 key results or arguments

4 conclusion

**Problem description:**

Codeforces 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>

Competitors participate in rated contests, then their rating will change. I will research the way people’s ratings change. Also, I would provide a strategy to get higher scores and push to higher ratings.

**Description of Codeforces contest mechanism**

Use official blogs to illustrate this part. It includes time penalty, submission penalty, successful hacks that increase scores, and unsuccessful hacks that decrease scores. The mechanism is a bit complicated.

**My ranking algorithm: Elo method that applies to one-versus-many** (I don’t know if there is this kind of algorithm. If there is then I can just adopt that name)

In competition, contestants have their scores for each problem. I hypothesize that the rating change of an individual is based on the following aspects:

Rating before the contest (very likely)

Difficulty of each problem: different difficulty of problems indicates different weights in score/rating computation. It can be defined by the problem givers or reflected by participants’ scores.

Score on each problem

Total score in the contest (very likely)

Ranking in the contest (very likely)

I will design a ranking algorithm: a modified version of Elo, referencing the official rating system of Codeforces at <https://codeforces.com/blog/entry/20762>. But this system was designed six years ago, so its method can only be as a reference. I may modify this method slightly to make it correspond to my data.

**Some experiments**

Next, I will do some experiments to test which aspects consist of a part of rating computation.

I will collect the data of 20 rated contests in different categories – different periods but within one year ago, and different divisions.

Then run the algorithm, check my result, and improve using the predictability index: using R-squared to measure the variance of the difference between my predicted rating change and the actual rating change.

After that, I will modify the method and re-run and repeat the process until I get a satisfactory result: hopefully improving R-squared to 95%

**A clever strategy of attaining higher scores and ratings in Codeforces**

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.

**Conclusion**

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.

**References**:

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

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

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

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

<http://codeforces.com/blog/entry/44214>（没啥用）

<https://www.luogu.com.cn/blog/ezoixx130/codeforces-advanced-tutorial（cf> 上分技巧）

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

Many websites introduce the Codeforces rating system, but I decided to create one on my own and their systems just as references. Maybe I will not need to use them in my essay.

Algorithm name: multi-competitor Elo

Or Elo-MMR: check 2101.00400.pdf

Trueskill code：<https://pypi.org/project/trueskill/>

Here is an article on a multi-competitor ranking with Elo  
<https://fivethirtyeight.com/features/formula-one-racing/>  
Another option is to work with a multi-competitor variation on the Elo method. The paper is a bit technical but is at:  
<http://www.glicko.net/research/multicompetitor.pdf>

**My actual paper:**

**An analysis of the Codeforces rating system**

Guang Yang

**1 Introduction**

Multi-competitor ranking is a present and on-going research area especially given the advent of massive online gaming. Microsoft Research proposed the TrueSkill method and TrueSkill 2, with TrueSkill 2 being their multi-competitor ranking method. < https://www.microsoft.com/en-us/research/publication/trueskill-2-improved-bayesian-skill-rating-system/ > There is also a version of Elo for multi-competitor games. One such version is <https://arxiv.org/pdf/2101.00400>. Other variations also exist such as the Bayesian approach introduced by <https://www.microsoft.com/en-us/research/publication/trueskill-2-improved-bayesian-skill-rating-system/>. In this paper, we 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 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>)

Competitors participate in rated contests, then their rating will change. I will research the way that competitors’ change of rating after the contest based on their performance in the contest.

**(Optional) Description of Codeforces contest mechanism**

The cost of the problem can be determined by the following table. For example, if the problem B was solved after 10 minutes of contest, then it costs 1000 — 4 \* 10 = 960 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 1000 — 4 \* 10 — 2 \* 50 = 860 points. However, no matter how much attempts you did and when you solved the problem, the cost of the solution can not drop below 30% of points (see the column "Min. Score (30 %)"). (<https://codeforces.com/blog/entry/456>)

Successful hack: +100

Unsuccessful hack: -50

One extra submission: -50

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |
| С | 1500 | 450 | 6 points | 780 |
| D | 2000 | 600 | 8 points | 1040 |
| E | 2500 | 750 | 10 points | 1300 |

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

1 calculate expected win rate between any two competitors.

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

and

TBD

**4 Apply it on data and check result (TBD)**

I use <Chartier’s code> or <<https://github.com/QAQrz/Codeforces-Rating-System>> to get the result of 20 matches. Data includes contestants’ rankings, IDs, official old rating and new rating. Then run my code and store the difference of my method and official rating change.

The predictability index: using R-squared to measure the variance of the difference between my predicted rating change and the actual rating change.

After that, I will modify the method and re-run and repeat the process until I get a satisfactory result: hopefully improving R-squared to 95%

**5 Other features**

**(TBD)**

**6 Conclusions**

**(TBD)**

**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].

And other useful links:

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

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

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

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

[http://codeforces.com/blog/entry/44214 (not](http://codeforces.com/blog/entry/44214%20(not) useful)

<https://www.luogu.com.cn/blog/ezoixx130/codeforces-advanced-tutorial>

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