Checkmate, Climbers

Ranking Climbing Difficulty with the Elo Rating System

Cooper LaPorte Jonah Weinbaum Dylan Fridman

Abstract

We compute a database of the Mountain Project climbing catalog storing information of over $240,000$ climbing routes and their associated reviews and comments. Using public information from this website, we develop what might be the first climbing grading system based on empirical data and not on the subjective opinion of climbers.

1 Introduction

1.1 Climbing and Climbing grades

Despite there being no objective metric to determine how hard a given rock climbing route is to climb, climbers throughout history have used different grading systems to judge the difficulty of climbing routes. However, they all fundamentally rely on the climbers' opinions. When a route is climbed for the first time, the first ascensionist proposes a grade and, as time passes and different climbers try the route, a consensus is reached, and the grade is determined. Thus, climbing grades are completely subjective.

Although most climbers think of climbing grades as being relatively reliable, there are many reasons to question their reliability. For example, many times people talk about grades in the context of the area: they don't say that a climb is 5.8 but "Yosemite 5.8". This particular clarification tries to convey that the climb might be harder than what the number itself might communicate. Furthermore, it is common knowledge that the earlier in climbing history that a given climbing route was established, the "stiffer" it will be (i.e., harder for the given grade). Furthermore, there is such a diverse range of climbing styles that performing well in a given style doesn't mean that the skills will translate to a different style (consider crack climbing and friction slab as an example).

Given all of this, it is only natural to try to come up with a grading system that doesn't rely on human opinion but is determined entirely by looking at the empirical data: are climbers more likely to fall climbing route X than climbing route Y? Well, then route X might be harder than route Y. Although this simple idea only allows us to compare a pair of climbs, comparing *every* pair of routes gives us a linear ordering of climbs by difficulty, which is exactly what a grading system tries to accomplish.

1.2 Mountain Project

Mountain Project is a virtual guidebook with information about most climbing routes in North America. For any given climb uploaded to the website, there is a lot of data such as the difficulty of a climb, user reviews and comments, images, and in-depth descriptions. The site contains roughly $300,000$ routes, $250,000$ of which are located in the US.

One of the many features of Mountain Project is that it allows climbers to "tick" (log) a climb after they've done it. When they do so, they are given several options to further specify how well they did on that climb:

- **Solo**: done without a rope.
- **Onsight**: no falls, first go, no previous information about the route.
- **Onsight**: no falls, first, with previous information about the route.
- **Redpoint**: no falls.
- **Fell/Hung**: had to weigh the rope at least once throughout the climb.

Most of the people who tick their climbs on Mountain Project also allow these ticks to be public, which in turn means that we can scrape them. We then use this information to empirically determine if a given climb is harder than another one. For example, if there's a climb X that most people solo or onsight while climb Y is rarely "sent" (climbed without falls), then route Y is definitely harder than route X. In order to obtain a linear ordering from these pairwise comparisons, we use the Elo Rating System.

1.3 The Elo Rating System

In 1960, the United States Chess Federation updated its ranking system to that of a system developed by chess player and statistician Aprad Elo. The aptly named **Elo rating system** was designed to rank players based on statistical estimation. The system considers each player's performance as a random variable, Aprad made the guiding assumption that each such variable was distributed according to some normal distribution - that is, players may perform particularly exceptionally or poorly but in general, their performance closely matches the mean of the distribution. A player's performance relative to another player tells us who we would expect to win, and we can use such an expectation to detect upsets where an underdog player performs better than we anticipate. Formally, we define

$$\begin{aligned} R_1 &= \text{ Player 1's Rating} \\ R_2 &= \text{ Player 2's Rating} \end{aligned}$$

Where these ratings lie roughly in the range of 100-2400. All players are originally given some base rating (say 1400). Then we have that the expected score (where a score of 1 is a game win and 0 is a game loss) of a player for a given game is

$$E_1 = \frac{1}{1 + 10^{(\frac{R_1 - R_2}{400})}}$$

$$E_2 = \frac{1}{1 + 10^{(\frac{R_2 - R_1}{400})}}$$

When a player's score (denoted S_i) exceeds their expected score (E_i) we update their rating to correct for the fact that their rating was underestimating their performance. We update according to the following formula

$$R_i' = R_i + K(S_i - E_i) \tag{1}$$

Where K is a constant that adjusts the maximum amount a player's rating can update. In theory, with sufficient matches, a player rating converges such that their expected score E_i is truly reflective of their score in a given game.

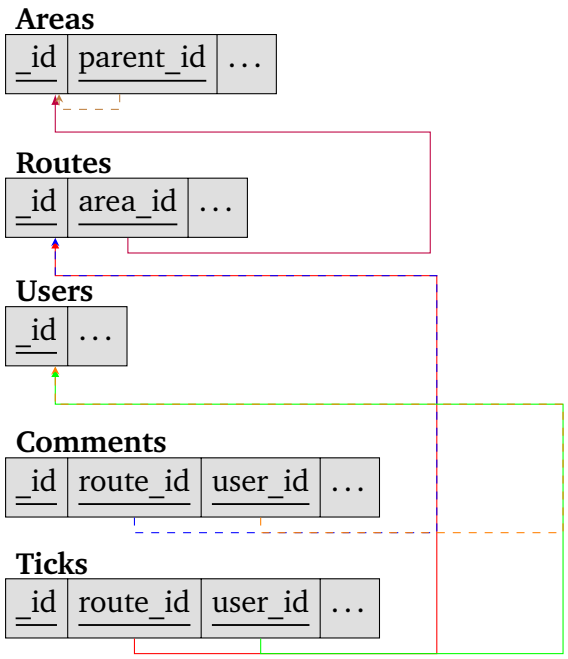
2 Methodology

2.1 Data Storage

Given the large amount of data being scraped, we wanted to ensure that data was backed up to a location that our whole project team could access. To do this, we built a Mongo NoSQL database on a server in the Trust Lab in the ECSC building. The database has a number of collections each pertaining to various components of website data including:

1. Areas - Areas point to other areas and ultimately to routes contained in areas
2. Routes - Route information including page text, descriptions, and difficulty
3. Users - Information including names of any users who have left a comment or tick on a route
4. Ticks - Information regarding the climbing style a user used on a route and whether or not they completed the route
5. Comments - General comments users left on a route to describe their thoughts

To illustrate how these collections reference each other we consider the following schema diagram



The database will be backed up after project completion but the Mongo server will not be maintained.

2.1.1 Accessing the Database

The database will be temporarily kept online for graders though at the grader's request an offline copy can be sent (it is simply to large to submit on canvas). To access the database through a Mongo shell, one must be on the *eduroam* network and can use the following connection string:

mongodb://EvilMonkey:&a@JREztYS5@EyPL@10.28.54.198:27017/?retryWrites=true&w=majority

2.2 Data Retreival

Data was scraped using methods from class, though at a much larger scale. This required efficient data processing as well as a great deal of parallelization, state tracking, and efficient database queries.

We first began by populating all routes and areas into our database by scraping through the route directory in a breadth-first search. We developed a state tracking system to quickly return to recently checked routes when the program crashed and this allowed us to (with lots of error checking) produce a full database of all routes in the United States.

We then indexed through each route, scraping comments and ticks and for each such entry examining the user who left the comment/tick and added them to our user database. Given that there are millions of ticks this was far too slow to compute without parallelization so we added a system of dividing routes among cores and tracking the state of all workers in case one failed. Even so, we were only able to scrape roughly 3500000 ticks with the time constraints

but this was sufficient to get a reasonable model that seemed to roughly track the difficulty level.

2.3 Ranking Routes

We consider climbing as a zero-sum game similar to chess. That is, two routes can play a match where the more difficult route wins. In this system, the performance of a route is directly tied to a route's difficulty. To accurately rank performance we use the Elo rating system as described above. Each route is given a base rating of $2,000$. We consider two routes to compete in a match whenever a user ticks both routes. Thus, we go through each user that has ticked climbed (in random order), and for each pair of routes they have ticked, we give each route a score from 0 to 4 that depends on how well the climber did on that route.

Climbing Style	**Route's score**
Solo	0
Onsight	1
Flash	2
Redpoint	3
Fell/Hung	4

If the climber did it in several styles, we take the lowest score. Then, we can get a number in $[0,1]$ that represents how much harder is route X than route Y (for this particular climber) by taking

$$S_X := \frac{t_X - t_Y + 4}{8}$$

where t_X is the score of route X and t_Y is the score of route Y. We can then use S_X to update the Elo ratings of route X as in Equation 1.

It is worth noting that we exclude from our final analysis routes that didn't get to play enough matches. Specifically, we only consider the final Elo ratings of routes that were ticked by more than 50 active users and we consider a user to be active if it has at least 10 ticks. We note that this rating scheme makes absolutely no use of the difficulty grade of the route thus making this a completely independent rating system.

2.4 Codebase Structure

The codebase is submitted along with this report on canvas but can additionally be found at the GitHub repository raikhen/scraping-mp [].

2.4.1 Utility Functions

Our codebase offers several utility functions in the *utils* folder. These include

1. *logger.py* : Handles data and error logging. Contains custom print functions that can be used to turn logging on or off. Logged statements are stored in a *logs* directory in the root folder. Each run of the program will have a unique identifier that will tie it to a specific log file.
2. *resume.py* : Handles returning to a former state when traversing the route directory. Given a route or area identifier, this will determine the original area from which the identifier originated.
3. *scrape.py* : Handles given a route identifier, these functions are able to extract route, tick and comment information as well as useful information about the overall route directory.
4. *grade_utils.py* : An enum used to quickly translate between route difficulty gradings and a simple numerical system.
5. *db_utils.py* : Handles connecting to the Mongo database as well as downloading the database for offline use.

2.4.2 Primary Functions

The main program files in the root folder are as follows

1. *config.py* : The file is simply used for stating whether values are logged, where they are logged, and whether the progress bar is used (though this fails in parallelized versions of functions).
2. *mp_db.py* : This handles the population of the database with areas, routes, users, ticks, and comments. It contains multiple versions of some functions some of which are parallel and some which are not.
3. *analysis.py* : This determines the correlation between our rating system and the rating system which Mountain Project uses.
4. *compute_elo.py* : This handles the ranking algorithm, examining and querying the database to perform matches between routes and update their relative rating.

3 Results

Since we only considered single-pitch rock routes and we further filtered routes to those that we could get a good estimate for, our final analysis uses a subset of 4454 routes. We obtain a correlation of ≈ 0.61 between the Elo rating and the climbing grade. Figure 1 allows us to further visualize the relation between the climbing grade and our computed Elo rating. These results could mean one of two things:

- **Climbing grades are very unreliable.** Although the correlation is far from insignificant, one can't retrieve the climbing grade from the Elo rating. Thus, it makes us think that the likelihood of someone falling on a climb doesn't actually correspond accurately to a climbing grade.

- **Our Elo rating system is too inaccurate.** It could be that requesting every route to play at least 500 matches (that correspond to at least 50 different climbers) is not strong enough of a requirement and that we need more data to get a better estimate of a climb's difficulty.

Regardless, this is still new territory in that, as far as we have found, this is the first objective climbing grading system. The fact that any correlation exists is somewhat surprising given that there was no consideration of the route difficulty grade in our Elo calculation.

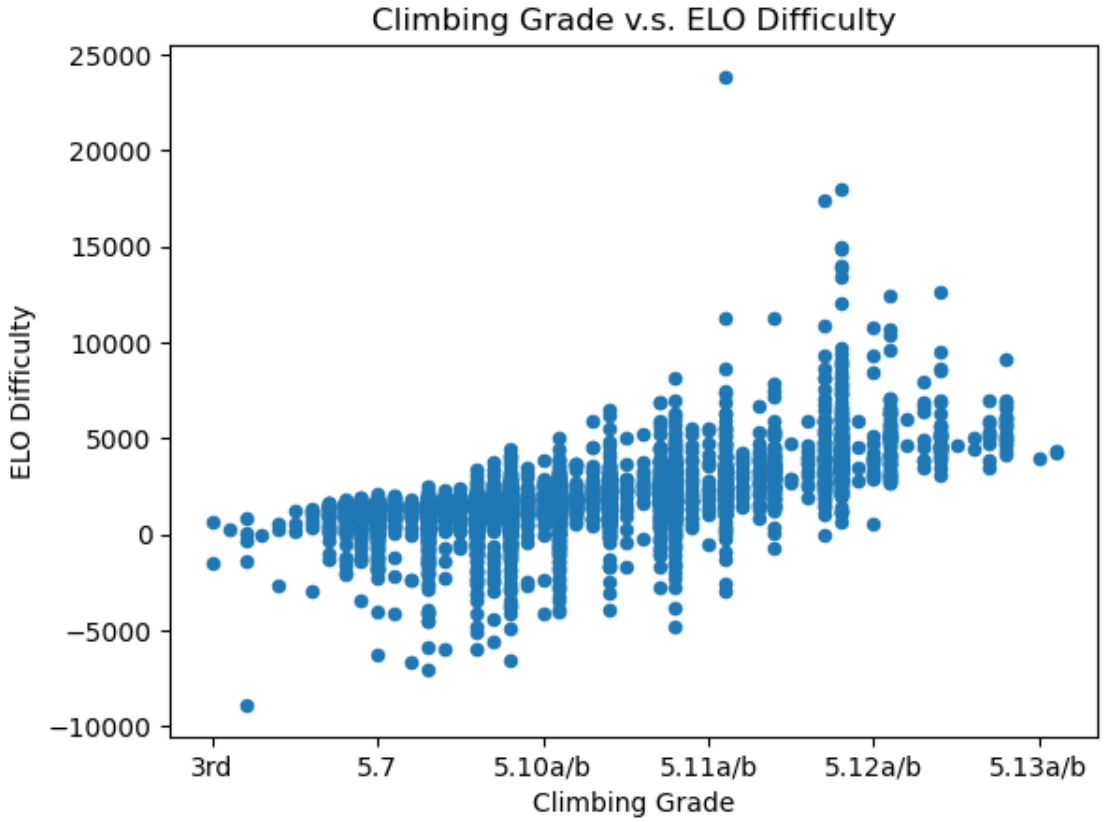


Figure 1. The relation between climbing grade and the Elo rating. Although we can distinguish a 5.7 from a 5.12d by the Elo rating, we cannot distinguish grades that are closer by.

4 Future Analysis

Due to time constraints, some difficulties in parallelization, and parsing issues, we weren't able to scrape every tick. In theory, this should create more matches which could help routes converge to a more objective difficulty level.

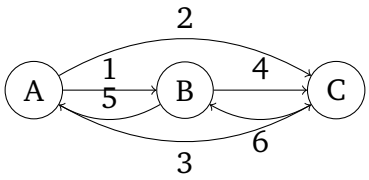
It would also be interesting to try slight variations on how we compute the Elo rating. It would be interesting to test if the Elo difficulty corresponds better to the climbing grade if we fix the number of matches each route can play or if we try different scoring functions (see *get_score* in *compute_elo.py*). We could also further break down climbers by time periods to account for

their improvement.

We had also discussed other zero-sum constructions of comparing routes. One option was to simultaneously rank climbers as well as routes by considering a route to "win" if a climber could not complete the route and to "lose" otherwise. There are potentially many other constructions that could more accurately rank route difficulty but this was meant more as a proof of concept and now that we have the database these further constructions would be much easier to test going forward if a future researcher wants to copy the database for testing.

Language sentiment analysis in comments could also be a further avenue for considering route difficulty. The huge amount of comments made this large-scale analysis intractable given the time left after scraping was completed.

Finally, this problem structure presents interesting generalizations in a graph streaming model which may be worth researching in its own right. That is, given a directed multigraph representing matches between nodes (where direction indicates who won a match), how can one, with space smaller than the number of nodes, query which route we expect to win given two routes? A simple example of this problem can be illustrated by the graph and stream below



with subsequent stream $\sigma = \langle 1, 5, 3, 6, 4, 2 \rangle$. The algorithm would then be asked to query whether A or C would be expected to win in a match, doing so without storing the ranking of every node. This problem could be generalized even further if we don't assume every node begins with the same base rating. After a discussion with Professor Amit Chakrabarti we suspect there may be a relationship between this problem structure and the vertex ordering problem on directed edge graph streams [1].

5 Conclusion

We believe that this empirical approach to climbing grades will eventually become the standard and we hope that this is a first step towards reaching that goal. We encourage others to use our database to compute statistics about routes across the United States as well as to create other models for objective rating systems.

References

[1] Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, page 1786–1802, 2020.

[2] Dylan Fridman, Jonah Weinbaum, and Cooper LaPorte. Scraping-mp. https://github.com/Raikhen/scraping-mp, 2023.