An Elo-like System for Massive Multiplayer Competitions

Aram Ebtekar
Vancouver, BC, Canada
aramebtech@gmail.com

Paul Liu
Stanford University
Stanford, CA, USA
paul.liu@stanford.edu

ABSTRACT

Skill estimation mechanisms, colloquially known as rating systems, play an important role in competitive sports and games. They provide a measure of player skill, which incentivizes competitive performances and enables balanced match-ups. In this paper, we present a novel Bayesian rating system for contests with many participants. It is widely applicable to competition formats with discrete ranked matches, such as online programming competitions, obstacle courses races, and video games. The system's simplicity allows us to prove theoretical bounds on its robustness and runtime. In addition, we show that it is *incentive-compatible*: a player who seeks to maximize their rating will never want to underperform. Experimentally, the rating system surpasses existing systems in prediction accuracy, and computes faster than existing systems by up to an order of magnitude.

CCS CONCEPTS

• **Information systems → Learning to rank**; • **Computing methodologies → Learning in probabilistic graphical models**.

KEYWORDS

rating system, skill estimation, mechanism design, competition, bayesian inference, robust, incentive-compatible, elo, glicko, trueskill

ACM Reference Format:
Aram Ebtekar and Paul Liu. 2021. An Elo-like System for Massive Multiplayer Competitions. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3442381.3450091

1 INTRODUCTION

Competitions, in the form of sports, games, and examinations, have been with us since antiquity. Many competitions grade performances along a numerical scale, such as a score on a test or a completion time in a race. In the case of a college admissions exam or a track race, scores are standardized so that a given score on two different occasions carries the same meaning. However, in events that feature novelty, subjectivity, or close interaction, standardization is difficult. The Spartan Races, completed by millions of runners, feature a variety of obstacles placed on hiking trails around the world [11]. Rock climbing, a sport to be added to the 2020 Olympics, likewise has routes set specifically for each competition. DanceSport, gymnastics, and figure skating competitions have a panel of judges who rank contestants against one another; these subjective scores are known to be noisy [32]. In all these cases, scores can only be used to compare and rank participants at the same event. Players, spectators, and contest organizers who are interested in comparing players' skill levels across different competitions will need to aggregate the entire history of such rankings. A strong player, then, is one who consistently wins against weaker players. To quantify skill, we need a **rating system**.

Good rating systems are difficult to create, as they must balance several mutually constraining objectives. First and foremost, rating systems must be accurate, in that ratings provide useful predictors of contest outcomes. Second, the ratings must be efficient to compute: within video game applications, rating systems are predominantly used for matchmaking in massively multiplayer online games (such as Halo, CounterStrike, League of Legends, etc.) [25, 29, 36]. These games have hundreds of millions of players playing tens of millions of games per day, necessitating certain latency and memory requirements for the rating system [12]. Third, rating systems must be **incentive-compatible**: a player's rating should never increase had they scored worse, and never decrease had they scored better. This is to prevent players from regretting a win, or from throwing matches to game the system. Rating systems that can be gamed often create disastrous consequences to player-base, potentially leading to the loss of players [3]. Finally, the ratings provided by the system must be human-interpretable: ratings are typically represented to players as a single number encapsulating their overall skill, and many players want to understand and predict how their performances affect their rating [21].

Classically, rating systems were designed for two-player games. The famous Elo system [18], as well as its Bayesian successors Glicko and Glicko-2, have been widely applied to games such as Chess and Go [21–23]. Both Glicko versions model each player's skill as a real random variable that evolves with time according to Brownian motion. Inference is done by entering these variables into the Bradley-Terry model [14], which predicts probabilities of game outcomes. Glicko-2 refines the Glicko system by adding a rating volatility parameter. Unfortunately, Glicko-2 is known to be flawed in practice, potentially incentivizing players to lose in what's known as "volatility farming". In some cases, these attacks can inflate a user's rating *several hundred points* above its natural value, producing ratings that are essentially impossible to beat via honest play. This was most notably exploited in the popular game of Pokemon Go [3]. See Section 5.1 for a discussion of this issue, as well as an application of this attack to the Topcoder rating system.

The family of Elo-like methods just described only utilize the binary outcome of a match. In settings where a scoring system provides a more fine-grained measure of match performance, Kovalchik [27] has shown variants of Elo that are able to take advantage of score information. For competitions consisting of several set

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450091

tasks, such as academic olympiads, Forišek [19] developed a model in which each task gives a different "response" to the player: the total response then predicts match outcomes. However, such systems are often highly application-dependent and hard to calibrate.

Though Elo-like systems are widely used in two-player settings, one needn't look far to find competitions that involve much more than two players. In response to the popularity of team-based games such as CounterStrike and Halo, many recent works focus on competitions that are between two teams [15, 24, 26, 28]. Another popular setting is many-player contests such as academic olympiads: notably, programming contest platforms such as Codeforces, Topcoder, and Kaggle [6, 8, 10]. As with the aforementioned Spartan races, a typical event attracts thousands of contestants. Programming contest platforms have seen exponential growth over the past decade, collectively boasting millions of users [5]. As an example, Codeforces gained over 200K new users in 2019 alone [2].

In "free-for-all" settings, where N players are ranked individually, the Bayesian Approximation Ranking (BAR) algorithm [34] models the competition as a series of $\binom{N}{2}$ independent two-player contests. In reality, of course, the pairwise match outcomes are far from independent. Thus, TrueSkill [25] and its variants [17, 29, 31] model a player's performance during each contest as a single random variable. The overall rankings are assumed to reveal the total order among these hidden performance variables, with various methods used to model ties and teams. For a textbook treatment of these methods, see [35]. These rating systems are efficient in practice, successfully rating userbases that number well into the millions (the Halo series, for example, has over 60 million sales since 2001 [4]).

The main disadvantage of TrueSkill is its complexity: originally developed by Microsoft for the popular Halo video game, TrueSkill performs approximate belief propagation, which consists of message passing on a factor graph, iterated until convergence. Aside from being less human-interpretable, this complexity means that, to our knowledge, there are no proofs of key properties such as runtime and incentive-compatibility. Even when these properties are discussed [29], no rigorous justification is provided. In addition, we are not aware of any work that extends TrueSkill to non-Gaussian performance models, which might be desirable to limit the influence of outlier performances (see Section 5.2).

It might be for these reasons that popular platforms such as Codeforces and Topcoder opted for their own custom rating systems. These systems are not published in academia and do not come with Bayesian justifications. However, they retain the formulaic simplicity of Elo and Glicko, extending them to settings with much more than two players. The Codeforces system includes ad hoc heuristics to distinguish top players, while curbing rampant inflation. Topcoder's formulas are more principled from a statistical perspective; however, it has a volatility parameter similar to Glicko-2, and hence suffers from similar exploits [19]. Despite their flaws, these systems have been in place for over a decade, and have more recently gained adoption by additional platforms such as CodeChef and LeetCode [1, 7].

Our contributions. In this paper, we describe the Elo-MMR rating system, obtained by a principled approximation of a Bayesian model similar to Glicko and TrueSkill. It is fast, embarrassingly parallel, and makes accurate predictions. Most interesting of all, its simplicity allows us to rigorously analyze its properties: the "MMR" in the name stands for "Massive", "Monotonic", and "Robust". "Massive" means that it supports any number of players with a runtime that scales linearly; "monotonic" is a synonym for incentive-compatible, ensuring that a rating-maximizing player always wants to perform well; "robust" means that rating changes are bounded, with the bound being smaller for more consistent players than for volatile players. Robustness turns out to be a natural byproduct of accurately modeling performances with heavy-tailed distributions, such as the logistic. TrueSkill is believed to satisfy the first two properties, albeit without proof, but fails robustness. Codeforces only satisfies incentive-compatibility, and Topcoder only satisfies robustness.

Experimentally, we show that Elo-MMR achieves state-of-the-art performance in terms of both prediction accuracy and runtime on industry datasets. In particular, we process the entire Codeforces database of over 400K rated users and 1000 contests in well under a minute, beating the existing Codeforces system by more than an order of magnitude while improving upon its accuracy. Furthermore, we show that the well-known Topcoder system is severely vulnerable to volatility farming, whereas Elo-MMR is immune to such attacks. A difficulty we faced was the scarcity of efficient open-source rating system implementations. In an effort to aid researchers and practitioners alike, we provide open-source implementations of all rating systems, dataset mining, and additional processing used in our experiments at `https://github.com/EbTech/Elo-MMR`.

We note that since releasing our preprint, Elo-MMR has already been put in production in industry settings [9].

Organization. In Section 2, we formalize the details of our Bayesian model. We then show how to estimate player skill under this model in Section 3, and develop some intuitions of the resulting formulas. As a further refinement, Section 4 models skill evolutions from players training or atrophying between competitions. This modeling is quite tricky as we choose to retain players' momentum while preserving incentive-compatibility. While our modeling and derivations occupy multiple sections, the system itself is succinctly presented in Algorithms 1 to 3. In Section 5, we perform a volatility farming attack on the Topcoder system and prove that, in contrast, Elo-MMR satisfies several salient properties, the most critical of which is incentive-compatibility. Finally, in Section 6, we present experimental evaluations, showing improvements over industry standards in both accuracy and speed.

2 A BAYESIAN MODEL FOR MASSIVE COMPETITIONS

We now describe the setting formally, denoting random variables by capital letters. A series of competitive **rounds**, indexed by $t = 1, 2, 3, \ldots$, take place sequentially in time. Each round has a set of participating **players** $\mathcal{P}_t$, which may in general overlap between rounds. A player's **skill** is likely to change with time, so we represent the skill of player i at time t by a real random variable $S_{i,t}$.

In round t, each player $i \in \mathcal{P}_t$ competes at some **performance** level $P_{i,t}$, typically close to their current skill $S_{i,t}$. The deviations $\{P_{i,t} - S_{i,t}\}_{i \in \mathcal{P}_t}$ are assumed to be i.i.d. and independent of $\{S_{i,t}\}_{i \in \mathcal{P}_t}$.

Performances are not observed directly; instead, a ranking gives the relative order among all performances $\{P_{i,t}\}_{i \in \mathcal{P}_t}$. In particular, ties are modelled to occur when performances are exactly equal, a zero-probability event when their distributions are continuous.[1] This ranking constitutes the observational **evidence** E_t for our Bayesian updates. The rating system seeks to estimate the skill $S_{i,t}$ of every player at the present time t, given the historical round rankings $E_{\leq t} := \{E_1, \ldots, E_t\}$.

We overload the notation Pr for both probabilities and probability densities: the latter interpretation applies to zero-probability events, such as in $\Pr(S_{i,t} = s)$. We also use colons as wildcards to denote collections of variables differing only in a subscript: for instance, $P_{:,t} := \{P_{i,t}\}_{i \in \mathcal{P}_t}$. The joint distribution described by our Bayesian model factorizes as follows:

$$\begin{aligned} &\Pr(S_{:,:}, P_{:,:}, E_:) \qquad (1)\\ &= \prod_i \Pr(S_{i,0}) \prod_{i,t} \Pr(S_{i,t} \mid S_{i,t-1}) \prod_{i,t} \Pr(P_{i,t} \mid S_{i,t}) \prod_t \Pr(E_t \mid P_{:,t}),\end{aligned}$$

$$\begin{aligned} \text{where } \Pr(S_{i,0}) &\text{ is the initial skill prior,}\\ \Pr(S_{i,t} \mid S_{i,t-1}) &\text{ is the skill evolution model (Section 4),}\\ \Pr(P_{i,t} \mid S_{i,t}) &\text{ is the performance model, and}\\ \Pr(E_t \mid P_{:,t}) &\text{ is the evidence model.}\end{aligned}$$

For the first three factors, we will specify log-concave distributions (see Definition 3.1). The evidence model, on the other hand, is a deterministic indicator. It equals one when E_t is consistent with the relative ordering among $P_{:,t}$, and zero otherwise.

Finally, our model assumes that the number of participants $|\mathcal{P}_t|$ is large. The main idea behind our algorithm is that, in sufficiently massive competitions, the evidence E_t contains enough information to infer very precise estimates for $P_{:,t}$. Hence, we can treat these performances as if they were observed directly.

With that in mind, we'll often discuss the distributions of variables whose round subscript is t, conditioned on either the **prior** context $P_{i,<t}$ or the **posterior** context $P_{i,\leq t}$: these are called prior and posterior distributions, respectively. In particular, suppose we have the skill prior:

$$\pi_{i,t}(s) := \Pr(S_{i,t} = s \mid P_{i,<t}). \qquad (2)$$

Now, we observe E_t. By Equation (1), it is conditionally independent of $S_{i,t}$, given $P_{i,\leq t}$. By the law of total probability,

$$\begin{aligned}&\Pr(S_{i,t} = s \mid P_{i,<t},\, E_t)\\ &= \int \Pr(S_{i,t} = s \mid P_{i,<t},\, P_{i,t} = p) \Pr(P_{i,t} = p \mid P_{i,<t},\, E_t)\, \mathrm{d}p.\end{aligned}$$

This integral is intractable in general, since the performance posterior $\Pr(P_{i,t} = p \mid P_{i,<t}, E_t)$ depends not only on player i, but also on our beliefs regarding the skills of all $j \in \mathcal{P}_t$. However, in the limit of infinite participants, Doob's consistency theorem [20] implies that the posterior concentrates at the true value $P_{i,t}$. That is, with probability one, as $|\mathcal{P}_t| \to \infty$,

$$\begin{aligned}&\Pr(S_{i,t} = s \mid P_{i,<t},\, E_t)\\ &\to \Pr(S_{i,t} = s \mid P_{i,\leq t}) \int \Pr(P_{i,t} = p \mid P_{i,<t},\, E_t)\, \mathrm{d}p\\ &= \Pr(S_{i,t} = s \mid P_{i,\leq t}).\end{aligned}$$

Since our posteriors are continuous, the convergence holds for all s simultaneously. Moreover, we don't even need the full evidence E_t. Let $E^L_{i,t} = \{j \in \mathcal{P} : P_{j,t} > P_{i,t}\}$ be the set of players against whom i lost, and $E^W_{i,t} = \{j \in \mathcal{P} : P_{j,t} < P_{i,t}\}$ be the set of players against whom i won. That is, we only look at who wins, draws, and loses against i. $P_{i,t}$ remains identifiable using only $(E^L_{i,t}, E^W_{i,t})$, which will be more convenient for our purposes.

In practice, we should care about the rate of convergence. Suppose we want our estimate to be within ε of $P_{i,t}$, with probability at least $1 - \delta$. By asymptotic normality of the posterior [20], it suffices to have $O(\frac{1}{\varepsilon^2} \log \frac{1}{\delta})$ participants. Experimentally, we see in Section 6.5 that Elo-MMR is competitive on all sizes of contests.

Bayesian ratings systems, such as Glicko and TrueSkill, make several simplifying assumptions to render their posterior updates tractable. Typically these are chosen ad hoc for convenience; however, having passed to a limit in which $P_{i,\leq t}$ is identified, our framework is able to rigorously justify such simplifications. Firstly, since $P_{i,\leq t}$ is a sufficient statistic for predicting $S_{i,t}$, it may be said that $(E^L_{i,\leq t}, E^W_{i,\leq t})$ are "almost sufficient" for $S_{i,t}$: any additional information, such as from domain-specific scoring systems, becomes redundant for the purposes of skill estimation. Secondly, conditioned on $P_{:,\leq t}$, the posterior skills $S_{:,t}$ are independent of one another. As a result, there are no inter-player correlations to model, and a player's posterior is unaffected by rounds in which they are not a participant. Finally, if we've truly identified $P_{i,t}$, then rounds later than t should not prompt revisions in our estimate for $P_{i,t}$. This obviates the need for expensive whole-history update procedures [16, 17], for the purposes of present skill estimation.[2]

Thus, when the initial prior, performance model, and evolution model are all Gaussian, treating $P_{i,t}$ as certain is the *only* simplifying approximation we will make; that is, in the limit $|\mathcal{P}_t| \to \infty$, our method performs *exact* inference on Equation (1). In the following sections, we focus some attention on generalizing the performance model to non-Gaussian log-concave families, parametrized by location and scale; here, a few minor approximations keep the derivations tractable. We will use the logistic distribution as a running example and see that it induces robustness; however, our framework is agnostic to the specific distributions used.

The prior **rating** $\mu^\pi_{i,t}$ and posterior rating $\mu_{i,t}$ of player i at round t should be statistics that summarize the player's prior and posterior skill distribution, respectively. We'll use the mode: thus, $\mu_{i,t}$ is the maximum a posteriori (MAP) estimate, obtained by setting s to maximize the posterior $\Pr(S_{i,t} = s \mid P_{i,\leq t})$. By Bayes' rule,

$$\begin{aligned}\mu^\pi_{i,t} &:= \arg\max_s \pi_{i,t}(s),\\ \mu_{i,t} &:= \arg\max_s \pi_{i,t}(s) \Pr(P_{i,t} \mid S_{i,t} = s). \qquad (3)\end{aligned}$$

This objective suggests a two-phase algorithm to update each player $i \in \mathcal{P}_t$ in response to the results of round t. In phase one, we estimate $P_{i,t}$ from $(E^L_{i,t}, E^W_{i,t})$. By Doob's consistency theorem, our estimate is extremely precise when $|\mathcal{P}_t|$ is large, so we assume it to be exact. In phase two, we update our posterior for $S_{i,t}$ and the rating $\mu_{i,t}$ according to Equation (3).

[1] The relevant limiting procedure is to treat performances within ε-width buckets as ties, and letting $\varepsilon \to 0$. This technicality appears in the proof of Theorem 3.2.

[2] As opposed to *historical* skill estimation, which is concerned with $P(S_{i,t} \mid P_{i,\leq t'})$ for $t' > t$. Whole-history methods can take advantage of future information.

3 SKILL ESTIMATION IN TWO PHASES

3.1 Performance estimation

In this section, we describe the first phase of Elo-MMR. For notational convenience, we assume all probability expressions to be conditioned on the prior context $P_{i,<t}$, and omit the subscript t.

Our prior belief on each player's skill S_i implies a prior distribution on P_i. Let's denote its probability density function (pdf) by

$$f_i(p) := \Pr(P_i = p) = \int \pi_i(s) \Pr(P_i = p \mid S_i = s)\,\mathrm{d}s, \qquad (4)$$

where $\pi_i(s)$ was defined in Equation (2). Let

$$F_i(p) := \Pr(P_i \le p) = \int_{-\infty}^{p} f_i(x)\,\mathrm{d}x,$$

be the corresponding cumulative distribution function (cdf). We'll also define the following functions, which will be associated with losses, draws, and wins, respectively:

$$\begin{aligned} l_i(p) &:= \frac{\mathrm{d}}{\mathrm{d}p} \ln(1 - F_i(p)) = \frac{-f_i(p)}{1 - F_i(p)}, \\ d_i(p) &:= \frac{\mathrm{d}}{\mathrm{d}p} \ln f_i(p) = \frac{f_i'(p)}{f_i(p)}, \\ v_i(p) &:= \frac{\mathrm{d}}{\mathrm{d}p} \ln F_i(p) = \frac{f_i(p)}{F_i(p)}. \end{aligned}$$

Evidently, $l_i(p) < 0 < v_i(p)$. Now we define what it means for the deviation $P_i - S_i$ to be log-concave.

Definition 3.1. *An absolutely continuous random variable on a convex domain is **log-concave** if its probability density function f is positive on its domain and satisfies*

$$f(\theta x + (1-\theta)y) > f(x)^{\theta} f(y)^{1-\theta}, \ \forall \theta \in (0,1), x \ne y.$$

Log-concave distributions appear widely, and include the Gaussian and logistic distributions used in Glicko, TrueSkill, and many others. We'll see inductively that our prior π_i is log-concave at every round. Since log-concave densities are closed under convolution [13], the independent sum $P_i = S_i + (P_i - S_i)$ is also log-concave. Log-concavity is made very convenient by the following lemma, proved in the appendix:

Lemma 3.1. *If f_i is continuously differentiable and log-concave, then the functions l_i, d_i, v_i are continuous, strictly decreasing, and*

$$l_i(p) < d_i(p) < v_i(p) \text{ for all } p.$$

For the remainder of this section, we fix the analysis with respect to some player i. As argued in Section 2, P_i concentrates very narrowly in the posterior. Hence, we can estimate P_i by its MAP, choosing p so as to maximize:

$$\Pr(P_i = p \mid E_i^L, E_i^W) \propto f_i(p) \Pr(E_i^L, E_i^W \mid P_i = p).$$

Define $j \succ i$, $j \prec i$, $j \sim i$ as shorthand for $j \in E_i^L$, $j \in E_i^W$, $j \in \mathcal{P} \setminus (E_i^L \cup E_i^W)$ (that is, $P_j > P_i$, $P_j < P_i$, $P_j = P_i$), respectively. The following theorem yields our MAP estimate:

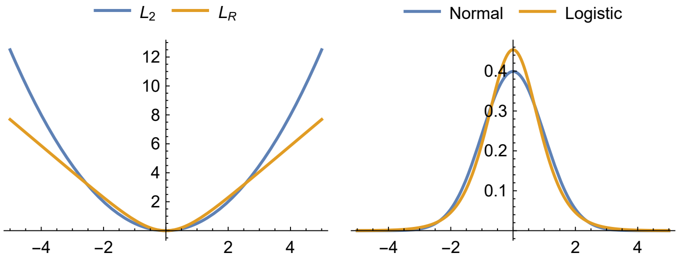

Figure 1: L_2 versus L_R for typical values (left). Gaussian versus logistic probability density functions (right).

Theorem 3.2. *Suppose that for all j, f_j is continuously differentiable and log-concave. Then the maximizer of* $\Pr(P_i = p \mid E_i^L, E_i^W)$ *is unique and given by the unique zero of*

$$Q_i(p) := \sum_{j \succ i} l_j(p) + \sum_{j \sim i} d_j(p) + \sum_{j \prec i} v_j(p).$$

The proof appears in the appendix. Intuitively, we're saying that the performance is the balance point between appropriately weighted wins, draws, and losses. Let's look at two specializations of our general model, to serve as running examples in this paper.

Gaussian performance model. If both S_j and $P_j - S_j$ are assumed to be Gaussian with known means and variances, then their independent sum P_j will also be a known Gaussian. It is analytic and log-concave, so Theorem 3.2 applies.

We substitute the well-known Gaussian pdf and cdf for f_j and F_j, respectively. A simple binary search, or faster numerical techniques such as the Illinois algorithm or Newton's method, can be employed to solve for the unique zero of Q_i.

Logistic performance model. Now we assume the performance deviation $P_j - S_j$ has a logistic distribution with mean 0 and variance β^2. In general, the rating system administrator is free to set β differently for each contest. Since shorter contests tend to be more variable, one reasonable choice might be to make $1/\beta^2$ proportional to the contest duration.

Given the mean and variance of the skill prior, the independent sum $P_j = S_j + (P_j - S_j)$ would have the same mean, and a variance that's increased by β^2. Unfortunately, we'll see that the logistic performance model implies a form of skill prior from which it's tough to extract a mean and variance. Even if we could, the sum does not yield a simple distribution.

For experienced players, we expect S_j to contribute much less variance than $P_j - S_j$; thus, in our heuristic approximation, we take P_j to have the same form of distribution as the latter. That is, we take P_j to be logistic, centered at the prior rating $\mu_j^{\pi} = \arg\max \pi_j$, with variance $\delta_j^2 = \sigma_j^2 + \beta^2$, where σ_j will be given by Equation (8). This distribution is analytic and log-concave, so the same methods based on Theorem 3.2 apply.

Let's derive Q_i explicitly in this case, since it has a rather intuitive form. The logistic distribution with variance δ_j^2 has scale parameter

$\bar{\delta}_j := \frac{\sqrt{3}}{\pi}\delta_j$; its cdf and pdf are:

$$F_j(p) = \frac{1}{1+e^{-(p-\mu_j^\pi)/\bar{\delta}_j}} = \frac{1}{2}\left(1+\tanh\frac{p-\mu_j^\pi}{2\bar{\delta}_j}\right),$$

$$f_j(p) = \frac{e^{(p-\mu_j^\pi)/\bar{\delta}_j}}{\bar{\delta}_j\left(1+e^{(p-\mu_j^\pi)/\bar{\delta}_j}\right)^2} = \frac{1}{4\bar{\delta}_j}\operatorname{sech}^2\frac{p-\mu_j^\pi}{2\bar{\delta}_j}.$$

They satisfy two very convenient relations:

$$F_j'(p) = f_j(p) = F_j(p)(1-F_j(p))/\bar{\delta}_j,$$
$$f_j'(p) = f_j(p)(1-2F_j(p))/\bar{\delta}_j,$$

from which it follows that

$$d_j(p) = \frac{1-2F_j(p)}{\bar{\delta}} = \frac{-F_j(p)}{\bar{\delta}} + \frac{1-F_j(p)}{\bar{\delta}} = l_j(p) + v_j(p).$$

In other words, a tie counts as the sum of a win and a loss. This can be compared to the approach (used in Elo, Glicko, BAR, Topcoder, and Codeforces) of treating each tie as half a win plus half a loss.[3]

Finally, putting everything together:

$$\begin{aligned} Q_i(p) &= \sum_{j \succ i} l_j(p) + \sum_{j \sim i}\left(l_j(p) + v_j(p)\right) + \sum_{j \prec i} v_j(p) \\ &= \sum_{j \succeq i} l_j(p) + \sum_{j \preceq i} v_j(p) \\ &= \sum_{j \succeq i} \frac{-F_j(p)}{\bar{\delta}_j} + \sum_{j \preceq i} \frac{1-F_j(p)}{\bar{\delta}_j}. \end{aligned}$$

Our estimate for P_i is the zero of this expression. Its terms correspond to probabilities, weighted by $1/\bar{\delta}_j$, of losing and winning against each player j. Accordingly, we can interpret $\sum_{j \in \mathcal{P}}(1 - F_j(p))/\bar{\delta}_j$ as a weighted expected rank of a player whose performance is p. P_i can thus be viewed as the performance level at which one's expected rank would equal i's actual rank. While the Codeforces and Topcoder systems compute performance values in a similar manner, here we've derived the formula from Bayesian principles.

3.2 Belief update

Having estimated $P_{i,t}$ in the first phase, the second phase is more straightforward. Ignoring normalizing constants, Equation (3) tells us that the pdf of the skill posterior can be obtained as the pointwise product of the pdfs of the skill prior and the performance model. When both factors are differentiable and log-concave, then so is their product. Its maximum is the new rating $\mu_{i,t}$; let's see how to compute it for the same two specializations of our model.

Gaussian performance model. When the skill prior and performance model are Gaussian with known means and variances, multiplying their pdfs yields another known Gaussian. Hence, the posterior is compactly represented by its mean $\mu_{i,t}$, which coincides with the MAP and rating; and its variance $\sigma_{i,t}^2$, which is our **uncertainty** regarding the player's skill.

Logistic performance model. When the performance model is non-Gaussian, the pointwise product of pdfs does not simplify so easily. By Equation (3), each round contributes an additional factor to the belief distribution. In general, we allow it to consist of a collection of simple log-concave factors, one for each round in which player i has participated. Denote i's participation history by

$$\mathcal{H}_{i,t} := \{k \in \{1,\ldots,t\} : i \in \mathcal{P}_k\}.$$

Since the factors deal with only a single player, we'll omit the subscript i. Specializing to the logistic setting, each $k \in \mathcal{H}_t$ contributes a logistic factor to the posterior, with mean p_k and variance β_k^2. We still use a Gaussian initial prior, with mean and variance denoted by p_0 and β_0^2, respectively. Postponing the discussion of skill evolution to Section 4, for the moment we assume that $S_k = S_0$ for all k. The posterior pdf, up to normalization, is then

$$\begin{aligned} &\pi_0(s)\prod_{k \in \mathcal{H}_t} \Pr(P_k = p_k \mid S_k = s) \\ &\propto \exp\left(-\frac{(s-p_0)^2}{2\beta_0^2}\right)\prod_{k \in \mathcal{H}_t}\operatorname{sech}^2\left(\frac{\pi}{\sqrt{12}}\frac{s-p_k}{\beta_k}\right). \end{aligned} \tag{5}$$

Maximizing the posterior density amounts to minimizing its negative logarithm. Up to a constant offset, this is given by

$$L(s) := L_2\left(\frac{s-p_0}{\beta_0}\right) + \sum_{k \in \mathcal{H}_t} L_R\left(\frac{s-p_k}{\beta_k}\right),$$

$$\text{where } L_2(x) := \frac{1}{2}x^2 \text{ and } L_R(x) := 2\ln\left(\cosh\frac{\pi x}{\sqrt{12}}\right).$$

$$\text{Thus, } L'(s) = \frac{s-p_0}{\beta_0^2} + \sum_{k \in \mathcal{H}_t}\frac{\pi}{\beta_k\sqrt{3}}\tanh\frac{(s-p_k)\pi}{\beta_k\sqrt{12}}. \tag{6}$$

L' is continuous and strictly increasing in s, so its zero is unique: it is the MAP μ_t. Similar to what we did in the first phase, we can solve for μ_t with binary search or other root-solving methods.

Furthermore, Equation (6) reveals a rather intuitive interpretation for the rating μ_t as an aggregate of the historical performances $p_{\leq t}$: Gaussian factors in L become L_2 penalty terms, whereas logistic factors appear as the more interesting L_R terms. In Figure 1, we see that L_R behaves quadratically near the origin, but linearly at the extremities. It's essentially a smoothed Huber loss, interpolating between L_2 and L_1 over a scale of magnitude β_k .

It is well-known that minimizing a sum of L_2 terms pushes the argument towards a weighted mean, while minimizing a sum of L_1 terms pushes the argument towards a weighted median. With L_R terms, the net effect is that μ_t acts like a robust average of the historical performances $p_{\leq t}$. Specifically, one can check that

$$\mu_t = \frac{\sum_k w_k p_k}{\sum_k w_k}, \text{ where } w_0 := \frac{1}{\beta_0^2} \text{ and}$$

$$w_k := \frac{\pi}{(\mu_t - p_k)\beta_k\sqrt{3}}\tanh\frac{(\mu_t - p_k)\pi}{\beta_k\sqrt{12}} \text{ for } k \in \mathcal{H}_t. \tag{7}$$

w_k is close to $1/\beta_k^2$ for typical performances, but can be up to $\pi^2/6$ times more as $|\mu_t - p_k| \to 0$, or vanish entirely as $|\mu_t - p_k| \to \infty$. The latter feature is due to the thicker tails of the logistic distribution, as compared to the Gaussian, resulting in an algorithm

[3] Elo-MMR, too, can be modified to split ties into half win plus half loss. It's easy to check that Lemma 3.1 still holds if $d_j(p)$ is replaced by $w_l l_j(p) + w_v v_j(p)$, provided that $w_l, w_v \in [0,1]$ and $|w_l - w_v| < 1$. In particular, we can set $w_l = w_v = 0.5$. The results in Section 5 won't be altered by this change.

that resists drastic rating changes in the presence of a few unusually good or bad performances. We'll formally state this *robustness* property in Theorem 5.7.

Estimating skill uncertainty. While there is no easy way to compute the variance of a posterior in the form of Equation (5), it will be useful to have some estimate σ_t^2 of uncertainty. There is a simple formula in the case where all factors are Gaussian. Since moment-matched logistic and normal distributions are relatively close (cf. Figure 1), we apply the same formula:

$$\frac{1}{\sigma_t^2} := \sum_{k \in \{0\} \cup \mathcal{H}_t} \frac{1}{\beta_k^2}. \tag{8}$$

3.3 Team competitions

While our main focus is on ranked competitions between a large number of *individuals*, Elo-MMR can be adapted to ranked competitions between a large number of *teams*. In this setting, round t's set of participants $\mathcal{P}_t$ is partitioned into a disjoint union of teams $\tau \in \mathcal{T}_t$: formally, $\mathcal{P}_t = \bigsqcup_{\tau \in \mathcal{T}_t} \tau$.

Instead of ranking individual i by their performance P_i, the competition ranks an entire team τ by a performance variable P_τ, which depends on the skills $\{S_i : i \in \tau\}$ of all its members. In general, the probabilistic team performance model should be domain-specific: depending, for instance, on whether game outcomes are most heavily influenced by a team's weakest or strongest player. A default choice that credits team members equally is the sum of their individual performances:

$$P_\tau := \sum_{i \in \tau} P_i = \sum_{i \in \tau} S_i + \sum_{i \in \tau} (P_i - S_i).$$

Thus, P_τ is a sum of $2|\tau|$ independently distributed terms. Just as before, we approximate this sum by a single Gaussian or logistic term with matching moments. Instead of the moments (μ_i^π, δ_i) of P_i in Algorithm 1, we'll have

$$\mu_\tau^\pi \leftarrow \sum_{i \in \tau} \mu_i,$$

$$\delta_\tau \leftarrow \sqrt{|\tau|\beta^2 + \sum_{i \in \tau} \sigma_i^2}.$$

With this change, the algorithm proceeds almost exactly as before, with the performance estimation step operating at the level of teams instead of individuals, $P_\tau, \mu_\tau^\pi, \delta_\tau$ replacing P_i, μ_i^π, δ_i.

The main caveat is that, in our limit of large competitions, we only obtain precise estimates of the *team* performance P_τ. To estimate the *individual* performance P_i, which in turn approximates S_i, we subtract all of i's teammates' ratings from P_τ. Since

$$S_i = P_\tau - \sum_{j \in \tau, j \neq i} S_j - \sum_{j \in \tau} (P_j - S_j),$$

the variance of this estimate is not β^2, but $|\tau|\beta^2 + \sum_{j \in \tau, j \neq i} \sigma_j^2$. Since we don't know who to credit for a team outcome, it's impossible to precisely estimate P_i. As a result, the independence argument in Section 2 ceases to hold. Nonetheless, Elo-MMR for team contests continues to enjoy the properties described in Section 5.

While smarter credit-assignment schemes may be considered in future work, one should be wary of the risk that such mechanisms may motivate players to seek credit, even at the expense of a team's overall performance. By simply distributing the credit equally, we ensure that every individual's incentive is perfectly aligned with winning as a team.

4 SKILL EVOLUTION OVER TIME

Over time, as a player trains or rests, a player's skill can change. If we model skill as a static variable, our system will eventually grow so confident in its estimate that it will refuse to admit substantial changes. To remedy this, we introduce a skill evolution model, so that in general $S_t \neq S_{t'}$ for $t \neq t'$. Rather than simply being equal to the previous round's posterior, now the skill prior at round t is given by

$$\pi_t(s) = \int \Pr(S_t = s \mid S_{t-1} = x) \Pr(S_{t-1} = x \mid P_{<t}) \, \mathrm{d}x. \tag{9}$$

The factors in the integrand are the skill evolution model and the previous round's posterior, respectively. Following other Bayesian rating systems (e.g., Glicko, Glicko-2, and TrueSkill [22, 23, 25]), we model the skill changes $S_t - S_{t-1}$ as independent zero-mean Gaussians. That is, $\Pr(S_t \mid S_{t-1} = x)$ is a Gaussian with mean x and some variance γ_t^2.

There is some flexibility in how γ_t is set. Glicko, in its original presentation, sets γ_t^2 proportionally to the time elapsed since the last update, corresponding to a continuous Brownian motion. Codeforces and Topcoder simply set γ_t to a constant when a player participates, and zero otherwise, corresponding to changes that are in proportion to how often the player competes. Now we are ready to complete the two specializations of our rating system.

Gaussian performance model. If the performance model and the prior on S_{t-1} are both Gaussian, then the posterior on S_{t-1} is also Gaussian. Since $S_t = S_{t-1} + (S_t - S_{t-1})$ is a sum of independent Gaussians, its prior is Gaussian as well. By induction, the skill belief distribution forever remains Gaussian. As we'll see in Section 5.2, this Gaussian specialization of the Elo-MMR framework lacks the R for robustness, so we call it Elo-MMχ.

Logistic performance model. After a player's first participation, the posterior in Equation (5) becomes non-Gaussian, rendering the integral in Equation (9) intractable.

A very simple approach would be to replace the full posterior in Equation (5) by a Gaussian approximation with mean μ_t (equal to the posterior MAP) and variance σ_t^2 (given by Equation (8)). Then, as in the previous case, the intractable integral specializes to a simple addition of Gaussian random variables.

With this approximation, no memory is kept of the individual performances P_t. Priors are simply Gaussian, while the pdf of a skill posterior is the product of two factors: the Gaussian prior, and a logistic factor corresponding to the latest performance. To ensure robustness (see Section 5.2), μ_t is computed as the arg max of this posterior *before* replacement by its Gaussian approximation. We call the rating system that takes this approach Elo-MMR(∞).

As the name implies, it turns out to be a limiting case of Elo-MMR(ρ). In the general setting with $\rho \in [0, \infty)$, we keep the full posterior from Equation (5). Since we cannot tractably compute the effect of a Gaussian diffusion, we seek a heuristic derivation of the next round's prior, retaining a form similar to Equation (5) while satisfying many of the same properties as the intended diffusion.

4.1 Desirable properties of a "pseudodiffusion"

We begin by listing some properties that our skill evolution algorithm, henceforth called a "pseudodiffusion", should satisfy. It will have a size parameter γ^2, analogous to the variance of a Gaussian diffusion. The first two properties are natural:

- *Incentive-compatibility.* First and foremost, the pseudodiffusion must not break the incentive-compatibility of our rating system. That is, a rating-maximizing player should never be motivated to lose on purpose (see Theorem 5.5).
- *Rating preservation.* The pseudodiffusion must not alter the arg max of the belief density. That is, the rating of a player should not change: $\mu_t^\pi = \mu_{t-1}$.

In addition, we borrow four properties of Gaussian diffusions:

- *Correct magnitude.* A pseudodiffusion of size γ^2 must increase the skill uncertainty, as measured by Equation (8), by γ^2.
- *Composability.* Two pseudodiffusions applied in sequence, first with size γ_1^2 and then with size γ_2^2, must have the same effect as a single pseudodiffusion of size $\gamma_1^2 + \gamma_2^2$.
- *Zero diffusion.* In the limit as $\gamma \longrightarrow 0$, the effect of a pseudodiffusion must vanish, i.e., not alter the belief distribution.
- *Zero uncertainty.* In the limit as $\sigma_{t-1} \longrightarrow 0$ (i.e., when the previous rating μ_{t-1} is a perfect estimate of S_{t-1}), our prior on S_t must become Gaussian with mean μ_{t-1} and variance γ^2. Finer-grained information regarding the history $P_{<t}$ must be erased.

In particular, Elo-MMR(∞) fails the *zero diffusion* property because it simplifies the belief distribution, even when $\gamma = 0$. In the proof of Theorem 4.1, we'll see that Elo-MMR(0) fails the *zero uncertainty* property. Thus, it is in fact necessary to have ρ strictly positive and finite. In Section 5.2, we'll come to interpret ρ as a kind of inverse momentum.

4.2 A heuristic pseudodiffusion algorithm

Each factor in the posterior (see Equation (5)) has a parameter β_k. Define a factor's **weight** to be $w_k := 1/\beta_k^2$, which by Equation (8) contributes to the **total weight** $\sum_k w_k = 1/\sigma_t^2$. Here, unlike in Equation (7), w_k does not depend on $|\mu_t - p_k|$.

Recall that the approximation step of Elo-MMR(∞) replaces all the logistic factors by a single Gaussian whose variance is chosen to ensure that the total weight is preserved. In addition, its mean is chosen to preserve the player's rating, given by the unique zero of Equation (6). Finally, the diffusion step of Elo-MMR(∞) increases the Gaussian's variance, and hence the player's skill uncertainty, by γ_t^2; this corresponds to a decay in the weight.

To generalize the idea, we interleave the two steps in a continuous manner. The approximation step becomes a **transfer step**: rather than replace the logistic factors outright, we take away equal fractions from each of their weights, and *place the sum of removed weights onto a new Gaussian factor.* In order for this operation to preserve ratings, the new factor must be centered at μ_{t-1}. Since Gaussian pdfs compose, the prior Gaussian factor can be combined with the new one. The diffusion step becomes a **decay step**, reducing each factor's weight by equal fractions (possibly different from the fractions in the transfer step), chosen such that the overall uncertainty is increased by γ_t^2.

Algorithm 1 Elo-MMR($\rho, \beta, \gamma, \mu_{init}, \sigma_{init}$)

```
for all rounds $t$ do
    $\mathcal{P}, \preceq, \succeq \leftarrow$ outcome of round $t$
    for all players $i \in \mathcal{P}$ in parallel do
        if $i$ has never competed before then
            $\mu_i, \sigma_i \leftarrow \mu_{init}, \sigma_{init}$
            $p_i, w_i \leftarrow [\mu_i], [1/\sigma_i^2]$
        diffuse($i$)
        $\mu_i^\pi, \delta_i \leftarrow \mu_i, \sqrt{\sigma_i^2 + \beta^2}$
    for all players $i \in \mathcal{P}$ in parallel do
        update($i$)
```

To make the idea precise, we generalize the posterior from Equation (5) with fractional **multiplicities** ω_k: the k'th factor is raised to the power ω_k. As a result, Equations (6) and (8) become:

$$L'(s) = \frac{\omega_0(s-p_0)}{\beta_0^2} + \sum_{k\in\mathcal{H}_t} \frac{\omega_k \pi}{\beta_k\sqrt{3}} \tanh\frac{(s-p_k)\pi}{\beta_k\sqrt{12}},$$
$$\frac{1}{\sigma_t^2} := \sum_{k\in\{0\}\cup\mathcal{H}_t} w_k, \quad \text{where } w_k := \frac{\omega_k}{\beta_k^2}. \tag{10}$$

For $\rho \in [0,\infty)$, the Elo-MMR(ρ) algorithm continuously and simultaneously performs transfer and decay, with transfer proceeding at ρ times the rate of decay. Of course, for $\rho = \infty$, the transfer is instantaneous and only the 0'th term survives. Holding β_k fixed, changes to ω_k can be described in terms of changes to w_k:

$$\frac{\mathrm{d}w_0}{\mathrm{d}t} = -r(t)w_0 + \rho r(t) \sum_{k\in\mathcal{H}_t} w_k,$$
$$\frac{\mathrm{d}w_k}{\mathrm{d}t} = -(1+\rho)r(t)w_k \quad \text{for } k \in \mathcal{H}_t,$$

where the arbitrary decay rate $r(t)$ can be eliminated by a change of variable $\mathrm{d}\tau = r(t)\mathrm{d}t$. The evolution from the end of round $t-1$ to the start of round t corresponds to some interval $\Delta\tau$, over which the total weight will have decayed by a factor $\kappa_t := e^{-\Delta\tau}$. Solving the differential equations yields the new weights, distinguished by their round t subscripts:

$$w_{0,t} = \kappa_t w_{0,t-1} + \left(\kappa_t - \kappa_t^{1+\rho}\right) \sum_{k\in\mathcal{H}_t} w_{k,t-1},$$
$$w_{k,t} = \kappa_t^{1+\rho} w_{k,t-1} \quad \text{for } k \in \mathcal{H}_t. \tag{11}$$

The *correct magnitude* property requires the uncertainty to increase from σ_{t-1}^2 to $\sigma_{t-1}^2 + \gamma_t^2$. By Equations (10) and (11),

$$\frac{1}{\sigma_{t-1}^2 + \gamma_t^2} = \sum_{k\in\{0\}\cup\mathcal{H}_t} w_{k,t} = \kappa_t \sum_{k\in\{0\}\cup\mathcal{H}_t} w_{k,t-1} = \frac{\kappa_t}{\sigma_{t-1}^2},$$

Solving for the decay factor:

$$\kappa_t = \left(1 + \frac{\gamma_t^2}{\sigma_{t-1}^2}\right)^{-1}.$$

Algorithm 1 details the full Elo-MMR(ρ) rating system. Each round of competition yields a set of participants $\mathcal{P}_t$, along with their rank-ordering. New players are initialized with a Gaussian prior. Changes in player skill are modeled by Algorithm 2; note

Algorithm 2 diffuse(i)

```
$\kappa \leftarrow (1 + \gamma^2/\sigma_i^2)^{-1}$
$w_G, w_L \leftarrow \kappa^\rho w_{i,0}, (1 - \kappa^\rho) \sum_{k \geq 0} w_{i,k}$
$p_{i,0} \leftarrow (w_G p_{i,0} + w_L \mu_i)/(w_G + w_L)$
$w_{i,0} \leftarrow \kappa(w_G + w_L)$
for all $k > 0$ do
    $w_{i,k} \leftarrow \kappa^{1+\rho} w_{i,k}$
$\sigma_i \leftarrow \sigma_i / \sqrt{\kappa}$
```

Algorithm 3 update(i)

```
$p \leftarrow \underset{x \in \mathbb{R}}{\text{zero of}} \sum_{j \leq i} \frac{1}{\delta_j} \left( \tanh \frac{x - \mu_j^\pi}{2\bar{\delta}_j} - 1 \right) + \sum_{j \geq i} \frac{1}{\delta_j} \left( \tanh \frac{x - \mu_j^\pi}{2\bar{\delta}_j} + 1 \right)$
$p_i$.push($p$)
$w_i$.push($1/\beta^2$)
$\mu_i \leftarrow \underset{x \in \mathbb{R}}{\text{zero of}}\ w_{i,0}(x - p_{i,0}) + \sum_{k > 0} \frac{w_{i,k}\beta^2}{\bar{\beta}} \tanh \frac{x - p_{i,k}}{2\bar{\beta}}$
```

how the updated Gaussian term blends its old value with the new Gaussian term created by the transfer process. The first phase of Algorithm 3 estimates P_t as the zero of a function of x. Finally, the second phase computes μ_t as the zero of another function.

The hyperparameters ρ, β, γ are domain-dependent, and can be set by standard hyperparameter search techniques. The system's invariance to translation and scale allows $\mu_{init}, \sigma_{init}$ to be set arbitrarily; a common choice is 1500, 350 [23]. For convenience, we assume β and γ are fixed and use the shorthand $\bar{\beta} := \frac{\sqrt{3}}{\pi}\beta$. Whereas our exposition used global round indices, here a subscript k corresponds to the k'th round in player i's participation history.

Theorem 4.1. *Algorithm 2 with $\rho \in (0, \infty)$ meets all of the properties listed in Section 4.1.*

Proof. We go through each of the six properties in order.

- *Incentive-compatibility.* This property will be stated in Theorem 5.5. To ensure that its proof carries through, the relevant facts to note here are that the pseudodiffusion algorithm ignores the performances p_k, and centers the transferred Gaussian weight at the rating μ_{t-1}, which is trivially monotonic in μ_{t-1}.
- *Rating preservation.* Recall that the rating is the unique zero of L' in Equation (10). To see that this zero is preserved, note that the decay and transfer operations multiply L' by constants (κ_t and κ_t^ρ, respectively), before adding the new Gaussian term, whose contribution to L' is zero at its center.
- *Correct magnitude.* Follows from our derivation for κ_t.
- *Composability.* Follows from *correct magnitude* and the fact that every pseudodiffusion follows the same differential equations.
- *Zero diffusion.* As $\gamma \to 0$, $\kappa_t \to 1$. Provided that $\rho < \infty$, we also have $\kappa_t^\rho \to 1$. Hence, for all $k \in \{0\} \cup \mathcal{H}_t$, $w_{k,t} \to w_{k,t-1}$.
- *Zero uncertainty.* As $\sigma_{t-1} \to 0$, $\kappa_t \to 0$. The total weight decays from $1/\sigma_{t-1}^2$, which becomes extremely large in this limit, to γ^2. Provided that $\rho > 0$, we also have $\kappa_t^\rho \to 0$, so these weights transfer in their entirety, leaving behind a Gaussian with mean μ_{t-1}, variance γ^2, and no additional history. □

5 THEORETICAL PROPERTIES

In this section, we see how the simplicity of the Elo-MMR formulas enables us to rigorously prove that the rating system is incentive-compatible, robust, and computationally efficient.

5.1 Incentive-compatibility

To demonstrate the need for incentive-compatibility, let's look at the consequences of violating this property in the Topcoder and Glicko-2 rating systems. These systems track a "volatility" for each player, which estimates the variance of their performances. A player whose recent performance history is more consistent would be assigned a lower volatility score, than one with wild swings in performance. The volatility acts as a multiplier on rating changes; thus, players with an extremely low or high performance will have their subsequent rating changes amplified.

While it may seem like a good idea to boost changes for players whose ratings are poor predictors of their performance, this feature has an exploit. By intentionally performing at a weaker level, a player can amplify future increases to an extent that more than compensates for the immediate hit to their rating. A player may even "farm" volatility by alternating between very strong and very weak performances. After acquiring a sufficiently high volatility score, the strategic player exerts their honest maximum performance over a series of contests. The amplification eventually results in a rating that exceeds what would have been obtained via honest play. This type of exploit was discovered in Glicko-2 as applied to the Pokemon Go video game [3]. Table 5.3 of [19] presents a milder violation in Topcoder competitions.

To get a realistic estimate of the severity of this exploit, we performed a simple experiment on the first five years of the Codeforces contest dataset (see Section 6.1). In Figure 2, we plot the rating evolution of the world's #1 ranked competitive programmer, Gennady Korotkevich, better known as `tourist`. In the *control* setting, we plot his ratings according to the Topcoder and Elo-MMR(1) systems. We contrast these against an *adversarial* setting, in which we have `tourist` employ the following strategy: for his first 45 contests, `tourist` plays normally (exactly as in the unaltered data). For his next 45 contests, `tourist` purposely falls to last place whenever his Topcoder rating is above 2975. Finally, `tourist` returns to playing normally for an additional 15 contests.

This strategy mirrors the Glicko-2 exploit documented in [3], and does not require unrealistic assumptions (e.g., we don't demand `tourist` to exercise very precise control over his performances). Compared to a consistently honest `tourist`, the volatility farming `tourist` ended up *523 rating points ahead* by the end of the experiment, with almost 1000 rating points gained in the last 15 contests alone. Transferring the same sequence of performances to the Elo-MMR(1) system, we see that it not only is immune to such volatility-farming attacks, but it also penalizes the dishonest strategy with a rating loss that decays exponentially once honest play resumes.

Recall that a key purpose of modeling volatility in Topcoder and Glicko-2 was to boost rating changes for inconsistent players. Remarkably, Elo-MMR achieves the same effect: we'll see in Section 5.2 that, for $\rho \in [0, \infty)$, Elo-MMR(ρ) also boosts changes to inconsistent players. And yet, we'll now prove that no strategic incentive for purposely losing exists in *any* version of Elo-MMR.

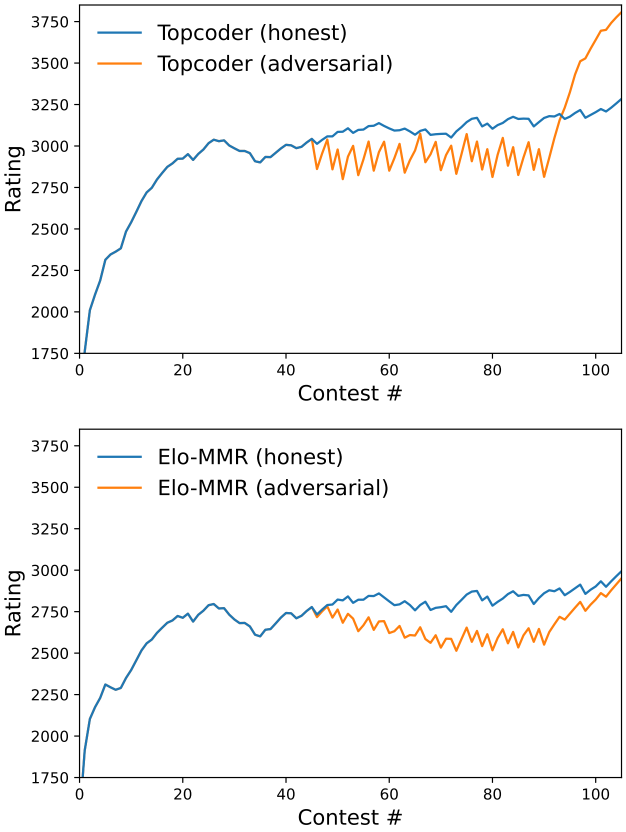

Figure 2: Volatility farming attack on the Topcoder system.

To this end, we need a few lemmas. Recall that, for the purposes of the algorithm, the performance p_i is defined to be the unique zero of the function $Q_i(p) := \sum_{j \succ i} l_j(p) + \sum_{j \sim i} d_j(p) + \sum_{j \prec i} v_j(p)$, whose terms l_j, d_j, v_j are contributed by opponents against whom i lost, drew, or won, respectively. Wins always contribute positively to a player's performance score, while losses contribute negatively:

Lemma 5.1. *Adding a win term to Q_i, or replacing a tie term by a win term, always increases its zero. Conversely, adding a loss term, or replacing a tie term by a loss term, always decreases it.*

Proof. By Lemma 3.1, $Q_i(p)$ is decreasing in p. Thus, adding a positive term will increase its zero whereas adding a negative term will decrease it. The desired conclusion follows by noting that, for all j and p,

$$\begin{aligned} v_j(p) &> 0, & v_j(p) - d_j(p) &> 0, \\ l_j(p) &< 0, & l_j(p) - d_j(p) &< 0. \end{aligned}$$

□

While not needed for our main result, a similar argument shows that performance scores are monotonic across the round standings:

Theorem 5.2. *If $i \succ j$ (that is, i beats j) in a given round, then the players' performance estimates satisfy $p_i > p_j$.*

Proof. If $i \succ j$ with i, j adjacent in the rankings, then

$$Q_i(p) - Q_j(p) = \sum_{k \sim i} (d_k(p) - l_k(p)) + \sum_{k \sim j} (v_k(p) - d_k(p)) > 0,$$

for all p. Since Q_i and Q_j are decreasing functions, it follows that $p_i > p_j$. By induction, the conclusion also holds for i, j that are not adjacent in the rankings. □

What matters for incentives is that performance scores be *counterfactually* monotonic; meaning, if we were to alter the round standings, a strategic player will always prefer to place higher:

Lemma 5.3. *In any given round, holding fixed the relative rankings among all players other than i (and holding fixed all preceding rounds), the performance p_i is a monotonic function of player i's prior rating and of player i's rank in this round.*

Proof. $Q_i(p)$ depends on the prior rating μ_i^π only through the self-tie term d_i, which in turn depends only on $p - \mu_i^\pi$. Thus, a change in μ_i^π has the same effect as an opposite change in p. By Lemma 3.1, d_i is monotonically *increasing* in μ_i^π, from which it follows that p_i is also monotonically increasing in μ_i^π.

Now, since an upward shift in i's ranking can only convert losses to ties and ties to wins, Lemma 5.1 implies that p_i is also monotonically increasing in improvements to i's rank. □

Having established the relationship between round rankings and performance scores, the next step is to prove that, even with hindsight, players will always prefer their performance scores to be as high as possible:

Lemma 5.4. *Holding fixed the set of contest rounds in which a player has participated, their current rating is monotonic in each of their past performance scores.*

Proof. The player's rating is given by the zero of L' in Equation (10). This expression contains the variables $\beta_:$, $\omega_:$, $p_:$, and s. As p_k is varied, $\beta_:$ and $\omega_:$ do not change: although the pseudodiffusions of Section 4 do modify $\omega_:$, these changes are agnostic to p_k. On the other hand, $L'(s)$ is monotonically increasing in s and decreasing in each of the p_k. Therefore, its zero is monotonically increasing in each of the p_k.

This is almost what we wanted to prove, except that p_0 is not a performance. Due to the pseudodiffusion's transfer step (or the actual diffusion, in the case of Elo-MMχ), p_0 is a weighted average of its previous value and the prior rating, and so it is monotonic in both. Using this same lemma in the previous round as an inductive hypothesis, it follows that p_0 is monotonic in past performances. By induction, the proof is complete. □

Finally, we conclude that a rating-maximizing player is always motivated to improve their round rankings:

Theorem 5.5 (Incentive-compatibility). *Holding fixed the set of contest rounds in which each player has participated, and the historical ratings and relative rankings among all players other than i, player i's current rating is monotonic in each of i's past rankings.*

Proof. Choose any contest round in player i's history, and consider improving player i's rank in that round while holding everything else fixed. It suffices to show that player i's current rating would necessarily increase as a result.

In the altered round, by Lemma 5.3, p_i is increased; and by Lemma 5.4, player i's post-round rating is increased. By Lemma 5.3 again, this increases player i's performance score in the following round. Proceeding inductively, we find that performance scores and ratings from this point onward are all increased. □

In the special cases of Elo-MMχ or Elo-MMR(∞), the rating system is "memoryless": the only data retained for each player are the current rating $\mu_{i,t}$ and uncertainty $\sigma_{i,t}$; detailed performance history is not saved. In this setting, we present a natural monotonicity theorem. A similar theorem was previously stated for the Codeforces system, albeit in an informal context without proof [8].

Theorem 5.6 (Memoryless Monotonicity). *In either the Elo-MMχ or Elo-MMR(∞) system, suppose i and j are two participants of round t. Suppose that the ratings and corresponding uncertainties satisfy $\mu_{i,t-1} \geq \mu_{j,t-1},\ \sigma_{i,t-1} = \sigma_{j,t-1}$. Then, $\sigma_{i,t} = \sigma_{j,t}$. Furthermore:*
If $i \succ j$ in round t, then $\mu_{i,t} > \mu_{j,t}$.
If $j \succ i$ in round t, then $\mu_{j,t} - \mu_{j,t-1} > \mu_{i,t} - \mu_{i,t-1}$.

Proof. The new contest round will add a rating perturbation with variance γ_t^2, followed by a new performance with variance β_t^2. As a result,

$$\sigma_{i,t} = \left(\frac{1}{\sigma_{i,t-1}^2 + \gamma_t^2} + \frac{1}{\beta_t^2} \right)^{-\frac{1}{2}} = \left(\frac{1}{\sigma_{j,t-1}^2 + \gamma_t^2} + \frac{1}{\beta_t^2} \right)^{-\frac{1}{2}} = \sigma_{j,t}.$$

The remaining conclusions are consequences of three properties: memorylessness, incentive-compatibility (Theorem 5.5), and translation-invariance (ratings, skills, and performances are quantified on a common interval scale relative to one another).

Since the Elo-MMχ or Elo-MMR(∞) systems are memoryless, we may replace the initial prior and performance histories of players with any alternate histories of our choosing, as long as our choice is compatible with their current rating and uncertainty. In particular, both i and j can be considered to have participated in the same set of rounds, with i always performing at $\mu_{i,t-1}$. and j always performing at $\mu_{j,t-1}$. Round t is unchanged.

Suppose $i \succ j$. Since i's historical performances are all equal or stronger than j's, Theorem 5.5 implies $\mu_{i,t} > \mu_{j,t}$.

Suppose $j \succ i$ instead. By translation-invariance, if we shift each of j's performances, up to round t and including the initial prior, upward by $\mu_{i,t-1} - \mu_{j,t-1}$, the rating changes between rounds will be unaffected. Players i and j now have identical histories, except that we still have $j \succ i$ at round t. Therefore, $\mu_{j,t-1} = \mu_{i,t-1}$ and, by Theorem 5.5, $\mu_{j,t} > \mu_{i,t}$. Subtracting the equation from the inequality proves the second conclusion. □

5.2 Robust response

Another desirable property in many settings is robustness: a player's rating should not change too much in response to any one contest, no matter how extreme their performance. The Codeforces and TrueSkill systems lack this property, allowing for unbounded rating changes. Topcoder achieves robustness by clamping any changes that exceed a cap, which is initially high for new players but decreases with experience.

When $\rho > 0$, Elo-MMR(ρ) achieves robustness in a natural, smoother manner. To understand how, we look at the interplay between Gaussian and logistic factors in the posterior. Recall the notation in Equation (10), describing the loss function and weights.

Theorem 5.7. *In the Elo-MMR(ρ) rating system, let*

$$\Delta_+ := \lim_{p_t \to +\infty} \mu_t - \mu_{t-1}, \quad \Delta_- := \lim_{p_t \to -\infty} \mu_{t-1} - \mu_t.$$

Then, for $\Delta_\pm \in \{\Delta_+, \Delta_-\}$,

$$\frac{\pi}{\beta_t \sqrt{3}} \left(w_0 + \frac{\pi^2}{6} \sum_{k \in \mathcal{H}_{t-1}} w_k \right)^{-1} \leq \Delta_\pm \leq \frac{\pi}{\beta_t \sqrt{3}} \frac{1}{w_0}.$$

Proof. The limits exist, by monotonicity. Using the fact that $0 < \frac{d}{dx} \tanh(x) \leq 1$, differentiating L' in Equation (10) yields

$$\forall s \in \mathbb{R},\ w_0 \leq L''(s) \leq w_0 + \frac{\pi^2}{6} \sum_{k \in \mathcal{H}_{t-1}} w_k.$$

Now, the performance at round t adds a new term with multiplicity one to $L'(s)$: its value is $\frac{\pi}{\beta_k \sqrt{3}} \tanh \frac{(s-p_k)\pi}{\beta_k \sqrt{12}}$. As a result, for every $s \in \mathbb{R}$, $\lim_{p_t \to \pm\infty} L'(s)$ increases by $\mp \frac{\pi}{\beta_t \sqrt{3}}$, while $\lim_{p_t \to \pm\infty} L''(s)$ does not change at all. Since we had $L'(\mu_{t-1}) = 0$ without this new term, after adding the term we have

$$\lim_{p_t \to \pm\infty} L'(\mu_{t-1}) \to \mp \frac{\pi}{\beta_t \sqrt{3}}.$$

Dividing by the former inequalities yields the desired result. □

The proof reveals that the magnitude of $\Delta_\pm$ depends inversely on that of L'' in the vicinity of the current rating, which in turn is related to the derivative of the tanh terms. If a player's performances vary wildly, the tanh terms will be widely dispersed, so any $s \in \mathbb{R}$ will necessarily be in the tail ends of most of the terms. Tails contribute very little to $L'(s)$, enabling a larger rating change. Conversely, the tanh terms of a player with a very consistent performance history will contribute large derivatives, so the bound on their rating change will be small.

Thus, Elo-MMR naturally caps the rating changes of all players, and the cap is smaller for consistent performers. The cap will increase after an extreme performance, providing a similar "momentum" to the Topcoder and Glicko-2 systems, but without sacrificing incentive-compatibility (Theorem 5.5).

Let's compare the lower and upper bound in Theorem 5.7: within a factor of $\pi^2/6$, their ratio corresponds to the normal term's weight w_0 relative to the total $\sum_k w_k$. Recall that ρ is the weight transfer rate: larger ρ results in more weight being transferred into w_0; in this case, the lower and upper bound tend to stay close together. Conversely, the momentum effect is more pronounced when ρ is small. In the extreme case $\rho = 0$, w_0 vanishes for experienced players, so a sufficiently volatile player would be subject to correspondingly large rating updates.

In general, according to Algorithm 2, the asymptotic steady-state values of w_0 and $W := \sum_k w_k$ must jointly solve the fixpoint equation

$$w_0 = \kappa w_0 + (\kappa - \kappa^{1+\rho})(W - w_0).$$

Rearranging yields an expression for the steady-state ratio:

$$\frac{w_0}{W} = \frac{\kappa - \kappa^{1+\rho}}{1 - \kappa^{1+\rho}}.$$

If we don't expect player skill to change too rapidly, then the system parameters should be set in such a way that $\kappa \approx 1$. In this limit, using $1 - \kappa^x \approx (1 - \kappa)x$ yields

$$\frac{w_0}{W} \approx \frac{(1-\kappa)\rho}{(1-\kappa)(1+\rho)} = \frac{1}{1+1/\rho}.$$

Thus, the upper bound in Theorem 5.7 is approximately proportional to $1+1/\rho$. Loosely speaking, therefore, the additive term $1/\rho$ may be interpreted as a momentum parameter.

5.3 Runtime analysis and optimizations

Let's look at the computation time needed to process a round with participant set $\mathcal{P}$, where we again omit the round subscript. Each player i has a participation history $\mathcal{H}_i$.

Estimating P_i entails finding the zero of a monotonic function with $O(|\mathcal{P}|)$ terms, and then obtaining the rating μ_i entails finding the zero of another monotonic function with $O(|\mathcal{H}_i|)$ terms. Using either of the Illinois or Newton methods, solving these equations to precision ε takes $O(\log\log\frac{1}{\varepsilon})$ iterations. As a result, the total runtime needed to process one round of competition is

$$O\left(\sum_{i\in\mathcal{P}}(|\mathcal{P}| + |\mathcal{H}_i|)\log\log\frac{1}{\varepsilon}\right).$$

This complexity is more than adequate for Codeforces-style competitions with thousands of contestants and history lengths up to a few hundred. Indeed, we were able to process the entire history of Codeforces on a small laptop in less than half an hour. Nonetheless, it may be cost-prohibitive in truly massive settings, where $|\mathcal{P}|$ or $|\mathcal{H}_i|$ number in the millions. Fortunately, it turns out that both functions may be compressed down to a bounded number of terms, with negligible loss of precision.

Adaptive subsampling. In Section 2, we used Doob's consistency theorem to argue that our estimate for P_i is consistent. Specifically, we saw that $O(1/\varepsilon^2)$ opponents are needed to get the typical error below ε. Thus, we can subsample the set of opponents to include in the estimation, omitting the rest. Random sampling is one approach. A more efficient approach chooses a fixed number of opponents whose ratings are closest to that of player i, as these are more likely to provide informative match-ups. On the other hand, if the setting requires incentive-compatibility to hold exactly, then one must avoid choosing different opponents for each player.

History compression. Similarly, it's possible to bound the number of stored factors in the posterior. Our skill-evolution algorithm decays the weights of old performances at an exponential rate. Thus, the contributions of all but the most recent $O(\log\frac{1}{\varepsilon})$ terms are negligible. Rather than erase the older logistic terms outright, we recommend replacing them with moment-matched Gaussian terms, similar to the transfers in Section 4 with $\kappa_t = 0$. Since Gaussians compose easily, a single term can then summarize an arbitrarily long prefix of the history.

Substituting $1/\varepsilon^2$ and $\log\frac{1}{\varepsilon}$ for $|\mathcal{P}|$ and $|\mathcal{H}_i|$, respectively, the runtime of Elo-MMR with both optimizations becomes

$$O\left(\frac{|\mathcal{P}|}{\varepsilon^2}\log\log\frac{1}{\varepsilon}\right).$$

Dataset	# contests	avg. # participants / contest
Codeforces	1257	3899
Topcoder	2115	391
Reddit	1000	20
CTF	1100	354
DanceSport	18292	6
Synth-large	50	10000
Synth-small	15000	5

Table 1: Summary of test datasets.

If the contests are *extremely large*, so that $\Omega(1/\varepsilon^2)$ opponents have a rating and uncertainty in the same ε-width bucket as player i, then it's possible to do even better: up to the allowed precision ε, the corresponding terms can be treated as duplicates. Hence, their sum can be determined by counting how many of these opponents win, lose, or tie against player i. Given the pre-sorted list of ranks of players in the bucket, two binary searches would yield the answer. In practice, a single bucket might not contain enough participants, so we sample enough buckets to yield the desired precision.

Simple parallelism. Since each player's rating computation is independent, the algorithm is embarrassingly parallel. Threads can read the same global data structures, so each additional thread contributes only $O(1)$ memory overhead.

6 EXPERIMENTS

In this section, we describe experiments on real-world datasets, mined from several sources that will be described in Section 6.1. We compare the rating systems described in Section 6.2, on the metrics of runtime and predictive accuracy, as described in Section 6.3. All experiments were run on a 2.3 GHz 8-core Skylake machine with 32 GB of memory. Implementations of all rating systems, dataset mining, and additional processing used in our experiments can be found at `https://github.com/EbTech/Elo-MMR`.

Hyperparameter search. To ensure fair comparisons, we ran a separate grid search for each triple of algorithm, dataset, and metric, over all of the algorithm's hyperparameters. The hyperparameter set that performed best on the first 10% of the dataset, was then used to test the algorithm on the remaining 90% of the dataset.

6.1 Datasets

Due to the scarcity of public domain datasets for rating systems, we mined five datasets to analyze the effectiveness of our system. The datasets were mined using data from each source website's inception up to February 5th, 2022. We also created synthetic datasets to test our system's performance when the data generating process matches our theoretical model. Summary statistics of the datasets are presented in Table 1.

Codeforces contest history. This dataset contains the current entire history of rated contests ever run on codeforces.com, the dominant platform for online programming competitions. The Codeforces platform has over 1 million registered users, over 400K of whom are rated, and has hosted over 1000 contests to date. Typically, each contest has a few thousand participants, takes 2 to 3 hours, and contains 5 to 8 problems. Players are ranked by total points, with more points typically awarded for tougher problems and for early solves. They may also attempt to “hack” one another's submissions for bonus points, identifying test cases that break their solutions.

Topcoder contest history. This dataset contains the current entire history of algorithm contests ever run on the topcoder.com. Topcoder is a predecessor to Codeforces, with over 1.4 million registered users, and a long history as a pioneering platform for programming contests. It hosts a variety of contest types, including over 2000 algorithm contests to date. The scoring system is similar to Codeforces, but with shorter rounds: typically 75 minutes allotted for a set of 3 problems.

SubredditSimulator threads. This dataset contains data scraped from the current top 1000 most upvoted threads on the website `reddit.com/r/SubredditSimulator`. Reddit is a social news aggregation website with over 400 million monthly active users. The site itself is broken down into sub-sites called subreddits. Users then post and comment to the subreddits, where the posts and comments receive votes from other users. In the subreddit SubredditSimulator, users are language generation bots trained on text from other subreddits. Automated posts are made by these bots to SubredditSimulator every 3 minutes, and real users of Reddit vote on the best bot. Each post (and its associated comments) can thus be interpreted as a round of competition between the bots who commented.

Capture the Flag competition history. This dataset contains data scraped from ctftime.org, an archive site for Capture the Flag (CTF) style computer security contests. Teams are scored based on the digital “flags" that they find by cracking computer security challenges. CTFtime tracks over 150K teams and 1000 competitions. Since these competitions are organized by a variety of groups, they come in a wide range of sizes.

DanceSport competition history. This dataset contains data scraped from results.o2cm.com. O^{2}CM is the dominant software package for hosting and managing competitive ballroom dance competitions in North America. Its freely accessible online database includes an average of one competition per week. Each competition is divided into events based on age category, syllabus level, and dance style. Since these events are judged and ranked separately, we process them as distinct rounds, in the order listed by O^{2}CM. Since modeling the chemistry between dance partners is beyond this paper's scope, we simply treat each dance couple as a distinct contestant.

Synthetic datasets (small and large). The small and large datasets contain 1K and 10K players respectively, with skills and performances generated according to the logistic generative model in Section 2. Players' initial skills are drawn i.i.d. with mean 1500 and variance 350^2. Players compete in all rounds, and are ranked according to independent performances with variance 200^2. Between rounds, we add i.i.d. Gaussian increments with variance 35^2 to each of their skills. In the small dataset, each round consists of just 5 players. In the large dataset, all 10K players participate in each round.

6.2 Rating systems

We compare our rating system against several academic and industry-tested alternatives. For a fairer comparison, we hand-coded efficient versions of all the algorithms in the safe subset of Rust, parellelized using the Rayon crate; as such, the Rust compiler verifies that they contain no data races [33]. The only exception is TrueSkill: the inherent seqentiality of its message-passing procedure prevented us from parallelizing it.

Elo-MMR. We specialize our rating system into two types: Elo-MMχ with a Gaussian performance model, and Elo-MMR(ρ) with a logistic performance model and pseudodiffusion rate ρ. We make use of the optimizations in Section 5.3, bounding both the number of sampled opponents and the history length by 500.

Topcoder system. The Topcoder website provides not only one of the oldest dataset of programming competitions, but also one of the oldest massively multiplayer deployments of a rating system. The Topcoder system [10] generalizes Glicko-2, and suffers from the same lack of incentive-compatibility [19]. Close variants of this system are used by other contest sites, such as CodeChef [1].

Codeforces system. In response to the main drawback of Topcoder, the Codeforces rating system [8] was specifically designed to be incentive-compatible. It features more ad hoc choices than the other systems: for instance, its rating updates target the geometric mean of a player's expected and actual ranks. Close variants of this system are used by other contest sites, such as LeetCode [7].

TrueSkill. We use the improved TrueSkill algorithm of [31], basing our code on an open-source implementation of the same algorithm. Developed for the purpose of video game matchmaking on Microsoft's Xbox Live platform, TrueSkill [25] is a Bayesian rating system, implemented using a powerful probabilistic programming framework. Its update rules are rather complex, requiring iterations of approximate message passing. It's very effective on games with moderate numbers of players (typically 2 to 16), but struggles in our experiments involving hundreds to thousands of players.

Glicko. The Glicko rating system [22] is a classic extension of Elo which, unlike Glicko-2, is incentive-compatible. While the Bayesian mathematics of Glicko was derived only for 2-player games, a naive baseline for N-player games can be obtained by decomposing the game into its N^2 pairwise matchups (including self-draws). Since these outcomes are far from independent, we normalize the collective weight of all N updates applying to each player, to match that of a hypothetical maximally informative 2-player game, i.e., against an equally skilled player whose skill is completely certain.

BAR. Bayesian Approximation Ranking [34] shares our goal of combining the accuracy of TrueSkill with the simplicity of Glicko. By a judicious application of simplifying approximations, it derives analytical formulas similar to the pairwise decomposition of Glicko[4]. The normalization in the original paper performs poorly on our datasets' large matches. To improve accuracy, just as with Glicko, we normalize the collective weight of the batched updates to equal that of one maximally informative 2-player game.

[4] Specifically, we use the Bradley-Terry model with full-pair, listed under Algorithm 1 in the source paper [34].

6.3 Evaluation metrics

To compare the different algorithms, we define two measures of predictive accuracy. Each metric will be defined on individual contestants in each round, and then averaged:

$$\textbf{aggregate}(\textbf{metric}) := \frac{\sum_t \sum_{i \in \mathcal{P}_t} \textbf{metric}(i,t)}{\sum_t |\mathcal{P}_t|}.$$

Pair inversion metric [25]. Our first metric computes the fraction of opponents against whom our ratings predict the correct pairwise result, defined as the higher-rated player either winning or tying:

$$\textbf{pair\_inversion}(i,t) := \frac{\text{\# correctly predicted matchups}}{|\mathcal{P}_t| - 1} \times 100\%.$$

This metric was used in the original evaluation of TrueSkill [25] and is related to the Kendall's τ rank correlation coefficient.

Rank deviation. Our second metric compares the rankings with the total ordering that would be obtained by sorting players according to their prior rating. The penalty is proportional to how much these ranks differ for player i:

$$\textbf{rank\_deviation}(i,t) := \frac{|\text{actual rank} - \text{predicted rank}|}{|\mathcal{P}_t| - 1} \times 100\%.$$

In the event of ties, among the ranks within the tied range, we use the one that comes closest to the rating-based prediction.

6.4 Empirical results

Recall that Elo-MMχ has a Gaussian performance model, matching the modeling assumptions of Topcoder and TrueSkill. Elo-MMR(ρ), on the other hand, has a logistic performance model, matching the modeling assumptions of Codeforces and Glicko. While ρ was included in the hyperparameter search, in practice we found that all values between 0 and 1 produce very similar results.

To ensure that errors due to the unknown skills of new players don't dominate our metrics, we excluded players who had competed in less than 5 total contests. In most of the datasets, this reduced the performance of our method relative to the others, as our method seems to converge more accurately. Despite this, we see in Table 2 that both versions of Elo-MMR outperform the other rating systems in both the pairwise inversion metric and the ranking deviation metric.

We highlight a few key observations. First, significant performance gains are observed on the Codeforces and Topcoder datasets, despite these platforms' rating systems having been designed specifically for their needs. Our gains are smallest on the synthetic dataset, for which all algorithms perform similarly. This might be in part due to the close correspondence between the generative process and the assumptions of these rating systems. Furthermore, the synthetic players compete in all rounds, enabling the system to converge to near-optimal ratings for every player. Finally, the improved TrueSkill performed well below our expectations, despite our best efforts to improve it. We suspect that the message-passing numerics break down in contests with a large number of individual participants. The difficulties persisted in all TrueSkill implementations that we tried, including on Microsoft's popular `Infer.NET` framework [30]. To our knowledge, we are the first to present experiments with TrueSkill on contests where the number of participants is in the hundreds or thousands. One case where TrueSkill outperformed is in the DanceSport dataset, where the average number of participants per contest is just 3. In preliminary experiments, TrueSkill and Elo-MMR score about equally when the number of ranks is less than about 60.

Now, we turn our attention to Table 3, which showcases the computational efficiency of Elo-MMR. On smaller datasets, it performs comparably to the Codeforces, TrueSkill, and Topcoder algorithms. However, the latter suffer from a quadratic time dependency on the number of contestants; as a result, Elo-MMR outperforms them by one to two orders of magnitude on the larger Codeforces dataset.

Finally, in comparisons between the two Elo-MMR variants, we note that while Elo-MMR(ρ) is more accurate, Elo-MMχ is always faster. This has to do with the skill drift modeling described in Section 4, as every update in Elo-MMR(ρ) must process $O(\log \frac{1}{\varepsilon})$ terms of a player's competition history.

6.5 Elo-MMR on small and large contests

The derivation in Section 2 depended on taking a limit in which the number of participants in each contest went to infinity. In practice, one might wonder how well Elo-MMR handles smaller contests. To find out, we simulate what would happen if each Codeforces contest was administered separately to smaller groups of contestants. That is, for every chosen contest size N, the participants of each contest are split into groups of at most N. Each group is placed in a round, and ranked according to their relative placement in the original contest.

In Figure 3, we see that Elo-MMR continues to beat the other systems, regardless of contest size.

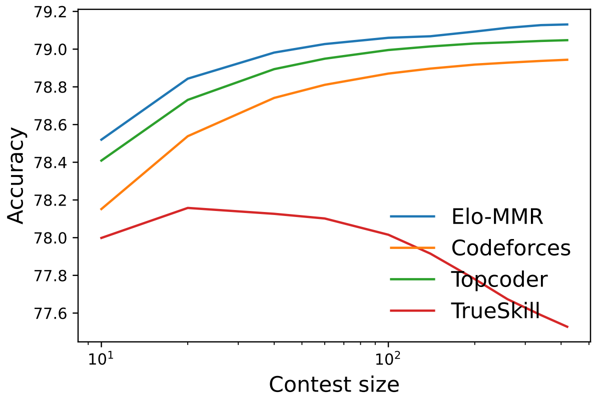

Figure 3: Number of participants vs. accuracy for various rating systems.

7 CONCLUSIONS

This paper introduces the Elo-MMR rating system, which is in part a generalization of the two-player Glicko system, allowing any number of players. By developing a Bayesian model and taking the limit as the number of participants goes to infinity, we obtained simple, human-interpretable rating update formulas. Furthermore, we saw that the algorithm is incentive-compatible, robust to extreme

Dataset	Codeforces		Topcoder		TrueSkill		Elo-MMχ		Elo-MMR(ρ)	
	pair inv.	rank dev.	pair inv.	rank dev.	pair inv.	rank dev.	pair inv.	rank dev.	pair inv.	rank dev.
Codeforces	78.9%	14.5%	**79.0%**	**14.4%**	70.5%	19.8%	**79.0%**	**14.4%**	**79.0%**	**14.4%**
Topcoder	72.8%	18.4%	72.5%	18.5%	70.2%	20.0%	**73.3%**	**18.1%**	73.2%	**18.1%**
Reddit	61.5%	27.3%	61.5%	27.3%	61.3%	**27.3%**	**61.6%**	**27.2%**	**61.6%**	**27.2%**
CTF	**71.1%**	**20.0%**	71.0%	20.1%	70.9%	20.2%	70.6%	20.4%	**71.1%**	**20.0%**
DanceSport	71.0%	26.0%	70.9%	26.2%	**73.0%**	**24.5%**	72.0%	25.0%	71.8%	25.6%
Synth-large	84.0%	11.1%	**84.1%**	**11.0%**	83.3%	11.6%	84.0%	11.1%	84.0%	11.1%
Synth-small	83.4%	15.2%	83.4%	15.2%	83.3%	15.3%	83.6%	15.0%	**83.7%**	**15.0%**

Table 2: Performance of each rating system on the pairwise inversion and rank deviation metrics. Bolded entries denote the best performances (highest pair inv. or lowest rank dev.) on each metric and dataset.

Dataset	CF	TC	TS	Elo-MMχ	Elo-MMR(ρ)
Codeforces	1298.3	455.8	260.7	**39.4**	47.6
Topcoder	26.8	**13.8**	61.0	13.9	15.3
Reddit	4.6	4.7	**4.2**	4.7	4.7
CTF	20.1	8.13	39.2	**7.4**	7.7
DanceSport	74.4	71.0	**66.6**	73.5	73.8
Synth-Large	10442.0	3024.0	320.3	42.6	**37.1**
Synth-Small	62.7	60.6	**56.0**	62.0	61.7

Table 3: Total compute time over entire dataset, in seconds.

performances, asymptotically fast, and embarrassingly parallel. To our knowledge, our system is the first to rigorously prove all these properties in a setting with more than two individually ranked players. In terms of practical performance, we saw that it outperforms existing industry systems in both prediction accuracy and computation speed.

This work can be extended in several directions. First, the choices we made in modeling ties, pseudodiffusions, teams, and opponent subsampling are by no means the only possibilities consistent with our Bayesian model of skills and performances. Second, it may be possible to further improve accuracy by fitting more flexible performance and skill evolution models to domain-specific data. Third, it would be useful to analyze convergence in realistic settings, where the Bayesian model is not completely accurate. In particular, controlling long-term rating inflation or deflation is a challenge, since we can't directly compare players at different times.

Over the past decade, online competition communities such as Codeforces have grown exponentially. As such, considerable work has gone into engineering scalable and reliable rating systems. Unfortunately, many of these systems have not been rigorously analyzed in the academic community. We hope that our paper and open-source release will open new explorations in this area.

ACKNOWLEDGEMENTS

The authors are indebted to Daniel Sleator and Danica J. Sutherland for initial discussions that helped inspire this work, and to Nikita Gaevoy for the open-source improved TrueSkill upon which our implementation is based. Experiments in this paper are funded by a Google Cloud Research Grant. The second author is supported by a VMWare Fellowship and the Natural Sciences and Engineering Research Council of Canada.

APPENDIX

Lemma 3.1. *If f_i is continuously differentiable and log-concave, then the functions l_i, d_i, v_i are continuous, strictly decreasing, and*

$$l_i(p) < d_i(p) < v_i(p) \text{ for all } p.$$

Proof. Continuity of F_i, f_i, f_i' implies that of l_i, d_i, v_i. It's known [13] that log-concavity of f_i implies log-concavity of both F_i and $1 - F_i$. As a result, l_i, d_i, and v_i are derivatives of strictly concave functions; therefore, they are strictly decreasing. In particular, each of

$$v_i'(p) = \frac{f_i'(p)}{F_i(p)} - \frac{f_i(p)^2}{F_i(p)^2}, \quad l_i'(p) = \frac{-f_i'(p)}{1 - F_i(p)} - \frac{f_i(p)^2}{(1 - F_i(p))^2},$$

are negative for all p, so we conclude that

$$d_i(p) - v_i(p) = \frac{f_i'(p)}{f_i(p)} - \frac{f_i(p)}{F_i(p)} = \frac{F_i(p)}{f_i(p)} v_i'(p) < 0,$$

$$l_i(p) - d_i(p) = -\frac{f_i'(p)}{f_i(p)} - \frac{f_i(p)}{1 - F_i(p)} = \frac{1 - F_i(p)}{f_i(p)} l_i'(p) < 0.$$

□

Theorem 3.2. *Suppose that for all j, f_j is continuously differentiable and log-concave. Then the unique maximizer of* $\Pr(P_i = p \mid E_i^L, E_i^W)$ *is given by the unique zero of*

$$Q_i(p) = \sum_{j \succ i} l_j(p) + \sum_{j \sim i} d_j(p) + \sum_{j \prec i} v_j(p).$$

Proof. First, we rank the players by their buckets according to $\lfloor P_j / \epsilon \rfloor$, and take the limiting probabilities as $\epsilon \to 0$:

$$\begin{aligned}
\Pr(\lfloor \frac{P_j}{\epsilon} \rfloor > \lfloor \frac{p}{\epsilon} \rfloor) &= \Pr(p_j \geq \epsilon \lfloor \frac{p}{\epsilon} \rfloor + \epsilon) \\
&= 1 - F_j(\epsilon \lfloor \frac{p}{\epsilon} \rfloor + \epsilon) \to 1 - F_j(p), \\
\Pr(\lfloor \frac{P_j}{\epsilon} \rfloor < \lfloor \frac{p}{\epsilon} \rfloor) &= \Pr(p_j < \epsilon \lfloor \frac{p}{\epsilon} \rfloor) \\
&= F_j(\epsilon \lfloor \frac{p}{\epsilon} \rfloor) \to F_j(p), \\
\frac{1}{\epsilon} \Pr(\lfloor \frac{P_j}{\epsilon} \rfloor = \lfloor \frac{p}{\epsilon} \rfloor) &= \frac{1}{\epsilon} \Pr(\epsilon \lfloor \frac{p}{\epsilon} \rfloor \leq P_j < \epsilon \lfloor \frac{p}{\epsilon} \rfloor + \epsilon) \\
&= \frac{1}{\epsilon} \left(F_j(\epsilon \lfloor \frac{p}{\epsilon} \rfloor + \epsilon) - F_j(\epsilon \lfloor \frac{p}{\epsilon} \rfloor) \right) \to f_j(p).
\end{aligned}$$

Let L_{jp}^ϵ, W_{jp}^ϵ, and D_{jp}^ϵ be shorthand for the events $\lfloor \frac{P_j}{\epsilon} \rfloor > \lfloor \frac{p}{\epsilon} \rfloor$, $\lfloor \frac{P_j}{\epsilon} \rfloor < \lfloor \frac{p}{\epsilon} \rfloor$, and $\lfloor \frac{P_j}{\epsilon} \rfloor = \lfloor \frac{p}{\epsilon} \rfloor$. respectively. These correspond to a player who performs at p losing, winning, and drawing against j, respectively, when outcomes are determined by ϵ-buckets. Then,

$$\begin{aligned}\Pr(E_i^W, E_i^L \mid P_i = p) &= \lim_{\epsilon \to 0} \prod_{j \succ i} \Pr(L_{jp}^\epsilon) \prod_{j \prec i} \Pr(W_{jp}^\epsilon) \prod_{j \sim i, j \neq i} \frac{\Pr(D_{jp}^\epsilon)}{\epsilon} \\ &= \prod_{j \succ i} (1 - F_j(p)) \prod_{j \prec i} F_j(p) \prod_{j \sim i, j \neq i} f_j(p), \\ \Pr(P_i = p \mid E_i^L, E_i^W) &\propto f_i(p) \Pr(E_i^L, E_i^W \mid P_i = p) \\ &= \prod_{j \succ i} (1 - F_j(p)) \prod_{j \prec i} F_j(p) \prod_{j \sim i} f_j(p),\end{aligned}$$

$$\frac{\mathrm{d}}{\mathrm{d}p} \ln \Pr(P_i = p \mid E_i^L, E_i^W) = \sum_{j \succ i} l_j(p) + \sum_{j \prec i} v_j(p) + \sum_{j \sim i} d_j(p) = Q_i(p).$$

Since Lemma 3.1 tells us that Q_i is strictly decreasing, it only remains to show that it has a zero. If the zero exists, it must be unique and it will be the unique maximum of $\Pr(P_i = p \mid E_i^L, E_i^W)$.

To start, we want to prove the existence of p^* such that $Q_i(p^*) < 0$. Note that it's not possible to have $f_j'(p) \geq 0$ for all p, as in that case the density would integrate to either zero or infinity. Thus, for each j such that $j \sim i$, we can choose p_j such that $f_j'(p_j) < 0$, and so $d_j(p_j) < 0$. Let $\alpha = -\sum_{j \sim i} d_j(p_j) > 0$.

Let $n = |\{j : j \prec i\}|$. For each j such that $j \prec i$, since $\lim_{p \to \infty} v_j(p) = 0/1 = 0$, we can choose p_j such that $v_j(p_j) < \alpha/n$. Let $p^* = \max_{j \preceq i} p_j$. Then,

$$\sum_{j \succ i} l_j(p^*) \leq 0, \quad \sum_{j \sim i} d_j(p^*) \leq -\alpha, \quad \sum_{j \prec i} v_j(p^*) < \alpha.$$

Therefore,

$$\begin{aligned}Q_i(p^*) &= \sum_{j \succ i} l_j(p^*) + \sum_{j \sim i} d_j(p^*) + \sum_{j \prec i} v_j(p^*) \\ &< 0 - \alpha + \alpha = 0.\end{aligned}$$

By a symmetric argument, there also exists some q^* for which $Q_i(q^*) > 0$. By the intermediate value theorem with Q_i continuous, there exists $p \in (q^*, p^*)$ such that $Q_i(p) = 0$, as desired. □

REFERENCES

[1] CodeChef Rating Mechanism. codechef.com/ratings

[2] Codeforces: Results of 2019. codeforces.com/blog/entry/73683

[3] Farming Volatility: How a major flaw in a well-known rating system takes over the GBL leaderboard. reddit.com/r/TheSilphRoad/comments/hwff2d/farming_volatility_how_a_major_flaw_in_a/

[4] Halo Xbox video game franchise: in numbers. telegraph.co.uk/technology/video-games/11223730/Halo-in-numbers.html

[5] Kaggle milestone: 5 million registered users! kaggle.com/general/164795

[6] Kaggle Progression System. kaggle.com/progression

[7] LeetCode New Contest Rating Algorithm. leetcode.com/discuss/general-discussion/468851/New-Contest-Rating-Algorithm-(Coming-Soon)

[8] Open Codeforces Rating System. codeforces.com/blog/entry/20762

[9] Ratings migrated to Elo-MMR. https://dmoj.ca/post/206-ratings-migrated-to-elo-mmr

[10] Topcoder Algorithm Competition Rating System. topcoder.com/community/competitive-programming/how-to-compete/ratings

[11] Why Are Obstacle-Course Races So Popular? theatlantic.com/health/archive/2018/07/why-are-obstacle-course-races-so-popular/565130/

[12] Sharad Agarwal and Jacob R. Lorch. 2009. Matchmaking for online games and other latency-sensitive P2P systems. In *SIGCOMM 2009*. 315–326.

[13] Mark Yuying An. 1997. Log-concave probability distributions: Theory and statistical testing. (1997).

[14] Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* (1952), 324–345.

[15] Shuo Chen and Thorsten Joachims. 2016. Modeling Intransitivity in Matchup and Comparison Data. In *WSDM 2016*. 227–236.

[16] Rémi Coulom. [n.d.]. Whole-history rating: A Bayesian rating system for players of time-varying strength. In *CG 2008*. Springer, 113–124.

[17] Pierre Dangauthier, Ralf Herbrich, Tom Minka, and Thore Graepel. 2007. TrueSkill Through Time: Revisiting the History of Chess. In *NeurIPS 2007*. 337–344.

[18] Arpad E. Elo. 1961. New USCF rating system. *Chess Life* (1961), 160–161.

[19] RNDr Michal Forišek. 2009. Theoretical and Practical Aspects of Programming Contest Ratings. (2009).

[20] David A Freedman. 1963. On the asymptotic behavior of Bayes' estimates in the discrete case. *The Annals of Mathematical Statistics* (1963), 1386–1403.

[21] Mark E Glickman. 1995. A comprehensive guide to chess ratings. *American Chess Journal* (1995), 59–102.

[22] Mark E Glickman. 1999. Parameter estimation in large dynamic paired comparison experiments. *Applied Statistics* (1999), 377–394.

[23] Mark E Glickman. 2012. Example of the Glicko-2 system. *Boston University* (2012), 1–6.

[24] Linxia Gong, Xiaochuan Feng, Dezhi Ye, Hao Li, Runze Wu, Jianrong Tao, Changjie Fan, and Peng Cui. 2020. OptMatch: Optimized Matchmaking via Modeling the High-Order Interactions on the Arena. In *KDD 2020*. 2300–2310.

[25] Ralf Herbrich, Tom Minka, and Thore Graepel. 2006. TrueSkill$^{\text{TM}}$: A Bayesian Skill Rating System. In *NeurIPS 2006*. 569–576.

[26] Tzu-Kuo Huang, Chih-Jen Lin, and Ruby C. Weng. 2006. Ranking individuals by group comparisons. In *ICML 2006*. 425–432.

[27] Stephanie Kovalchik. 2020. Extension of the Elo rating system to margin of victory. *Int. J. Forecast.* (2020).

[28] Yao Li, Minhao Cheng, Kevin Fujii, Fushing Hsieh, and Cho-Jui Hsieh. 2018. Learning from Group Comparisons: Exploiting Higher Order Interactions. In *NeurIPS 2018*. 4986–4995.

[29] Tom Minka, Ryan Cleven, and Yordan Zaykov. 2018. *TrueSkill 2: An improved Bayesian skill rating system*. Technical Report MSR-TR-2018-8. Microsoft.

[30] T. Minka, J.M. Winn, J.P. Guiver, Y. Zaykov, D. Fabian, and J. Bronskill. /Infer.NET 0.3. Microsoft Research Cambridge. http://dotnet.github.io/infer.

[31] Sergey I. Nikolenko, Alexander, and V. Sirotkin. 2010. Extensions of the TrueSkill TM rating system. In *In Proceedings of the 9th International Conference on Applications of Fuzzy Systems and Soft Computing*. 151–160.

[32] Jerneja Premelč, Goran Vučković, Nic James, and Bojan Leskošek. 2019. Reliability of judging in DanceSport. *Front. Psychol.* (2019), 1001.

[33] Josh Stone and Nicholas D Matsakis. The Rayon library (Rust Crate). crates.io/crates/rayon

[34] Ruby C. Weng and Chih-Jen Lin. 2011. A Bayesian Approximation Method for Online Ranking. *J. Mach. Learn. Res.* (2011), 267–300.

[35] John Michael Winn. 2019. *Model-based machine learning*.

[36] Lin Yang, Stanko Dimitrov, and Benny Mantin. 2014. Forecasting sales of new virtual goods with the Elo rating system. *RPM* (2014), 457–469.