DEIM2020 I4-2(day1 p60)

競技プログラミングコンテストにおける タスクの難易度のリアルタイム推定

渡邊　綾仁†　　田島　敬史††

† 京都大学工学部情報学科　〒 606–8501 京都府京都市左京区吉田本町
†† 京都大学大学院情報学研究科　〒 606–8501 京都府京都市左京区吉田本町
E-mail: †watanabe.a@dl.soc.i.kyoto-u.ac.jp, ††tajima@i.kyoto-u.ac.jp

あらまし　本論文では，競技プログラミングサイトのコンテストにおけるタスクの難易度をコンテスト時間中にリアルタイムで推定する手法を提案する．競技プログラミングサイトにおけるタスクの難易度推定のための既存手法としては，項目反応理論に基づいた手法が存在するが，この手法では，コンテスト終了後にしか難易度の判定ができない．そこで，本研究では，コンテスト途中でも，それまでにタスクに正答した参加者が正答するまでにかかった時間や参加者のレーティングの情報，レーティングの持つ性質に基づいてタスクの難易度を推定する手法を提案する．実際のコンテストデータを用いて擬似的にリアルタイムでタスクの難易度推定を行ったところ，提案手法の方がコンテストの開始時刻から早い段階で正解の値に近い値を推定できることが確認できた．

キーワード　競技プログラミング，時系列データ，項目反応理論，クラウドソーシング

1　はじめに

競技プログラミングコンテストでは，自身のコーディング能力や数学力，アルゴリズムとデータ構造についての知識力を競うために，全世界から多くの参加者が集まっている．コンテストで出題されるタスクは，さまざまな難易度のものが用意されており，コンテスト参加者は好きな順番で解くことができる．コンテスト中は，他の参加者の正答状況や解答にかかった時間を順位表の形で閲覧することは可能であるが，そこから各タスクの難易度が実際にどれくらいであるかをリアルタイムで想定することは容易ではない．

現在，多くの競技プログラミングコンテストのサイトでは，ユーザ登録をしている人が，いつでも過去のコンテストで実際に出題されたタスクにアクセスすることができるデータベースが存在する．これらのデータベースでは，コンテスト終了後に，実際のコンテスト参加者の正答状況から難易度を推定している．コンテスト参加者にはイロレーティング [1], [2] をベースとしたレーティングがついているため，多くの競技プログラミングコンテストサイトで，タスクの難易度を表す数値はこのレーティングの値をベースとしている．

コンテスト終了後に難易度を推定する手法としては，項目反応理論 [3] の考え方を応用し，ユーザのレーティングをある幅に区切り，階級ごとにタスクの正答率を算出し，ロジスティック関数でフィッティングを行って，正答率が 50%となる値を求める方法がある．この方法をそのままリアルタイムの推定に応用すると，コンテスト開始から間もない時刻では，レーティングの高い参加者のデータが集まってしまうため，難易度の推定値は，コンテスト終了後に推定する値よりもかなり高い値が出てしまう．また，コンテスト開催中は，いろいろな参加者の解答データが増えていくため，難易度の推定値が安定するためには時間がかかる．

コンテスト終了後に難易度を推定する既存の手法では，コンテスト参加者の各々のレーティングの値と，各タスクに正答したか否かの二値の情報にしか着目していない．また，タスクに正答したコンテスト参加者が，タスクの正答にかかった時間の情報も全く考慮されていない．リアルタイムでタスクの難易度を推定するためには，コンテスト終了後に得られる情報よりも限られた情報の中で推定を行わなければならないため，タスクに正答していない参加者の情報や，タスクに正答できた参加者がタスクに正答するまでにかかった時間の情報は活用するべきであると考えられる．

本研究では，イロレーティングの性質やコンテスト参加者のレーティングに加えて，コンテスト参加者が実際に正答するのにかかった時間のデータを用いたり，まだタスクに正答できていない参加者が後の時刻で正答する状況を考えることにより，リアルタイムでタスクの難易度を推定する手法を提案する．本研究の手法の有効性を評価するために，擬似的にリアルタイム推定を行った場合の出力値と，コンテスト終了後のコンテスト参加者の正答状況を全て用いて難易度推定を行った場合の難易度の推定値との比較を行う．

2　関連研究

この章では，時系列データの予測に関する研究，クラウドソーシングにおけるタスクの難易度推定に関する研究，イロレーティング，および項目応答理論におけるロジスティックモデルについて説明した後，競技プログラミングコンテストにおいてコンテスト終了後にタスクの難易度を推定する手法について述べる．

2.1 時系列データの予測に関する研究との関連

時系列データの予測に関しては，すでに数多くの研究がなされている [4], [5]．オンラインでの予測は，時間的に変化していく時系列データになり，これの終了時の値を予測できれば，オンラインの難易度推定が可能となる．競技プログラミングコンテストにおけるタスクの難易度のリアルタイム推定においても，コンテスト参加者の解答状況がコンテスト開始から時間とともに徐々に明らかになっていく点や，コンテスト終了後に計算されるタスクの難易度推定値を，コンテスト開催時間中に予測する点で，時系列データの予測と関連が深い．

単純な時系列データの予測モデルとして指数平滑法を説明する．時刻 t での予測値を y_t, 時刻 t での実測値を x_t, α をパラメータとすると，指数平滑法の式は，

$$\begin{aligned} y_t &= (1-\alpha)y_{t-1} + \alpha x_{t-1} \\ y_t &= y_{t-1} + \alpha(x_{t-1} - y_{t-1}) \end{aligned} \tag{1}$$

と表される．α は 0 から 1 の間で設定し，1 に近ければ誤差 $x_t - y_t$ の影響を強く受け入れ，0 に近ければ誤差の影響を小さくすることができる．(1) 式を n 時点前までの時刻 $t-n$ まで展開すると，

$$y_t = \sum_{i=1}^{n} \alpha(1-\alpha)^{i-1} x_{t-i} + (1-\alpha)^n y_{t-n} \tag{2}$$

となる．この (2) 式は，先行する n 個のデータについて，現在の時刻に近い値ほど大きい重みを与えることを意味している．指数平滑法では，先行する n 個の実測値を用いることによって次の時刻の値を予測することができる．

2.2 クラウドソーシングのタスクの難易度推定に関する研究

クラウドソーシングにおいては，多数決の結果の精度を向上させるために，タスクの難易度をモデル化した Whitehill らのモデル [6] や，タスクに対してのワーカーの得意不得意を考慮した Welinder らのモデル [7] が考案されている．

はじめに，Whitehill らのモデルの概要について説明する．ワーカー j の能力を α_j，i 番目のタスクの易しさを $\beta_i (> 0)$ とおくと，ワーカー j がタスク i に正答できる確率 P_{ij} は，

$$P_{ij} = \frac{1}{1 + \exp(-\alpha_j \beta_i)}$$

と表せる．P_{ij} の値が 1 に近づくほどワーカー j がタスク i に正答できる確率が高いことを表し，0 に近づくほど正答する確率が低いことを表している．ワーカー j の能力 α_j について，$\alpha_j = 0$ とすると $P_{ij} = 1/2$ となるので，このときちょうど 1/2 の確率でワーカー j がタスク i に正答できることになる．また，α_j の値が大きくなるほど，正答する確率は 1 に近づき，値が小さくなるほど正答する確率は 0 に近づく．タスク i の易しさ β_i については，β_i の値が大きくなるほどタスクの難易度が易しく，ワーカー j が正答できる確率が高くなることを表している．

次に，Welinder らのモデルの概要について説明する．Welinder らのモデルでは，ワーカーの能力とタスクをそれぞれ D 次元の実数値ベクトルで表すことによって，Whitehill らのモデルを拡張したモデルと考えることができる．ワーカー j の能力を実数値ベクトル $\boldsymbol{w}_j \in \mathbb{R}^D$，タスク i を実数値ベクトル $\boldsymbol{x} \in \mathbb{R}^D$ で表すことにする．また，各ワーカーは，各タスク i に対して，0 または 1 のいずれかのラベルをつけるものとし，閾値を $\theta \in \mathbb{R}$ とする．$\boldsymbol{w}_j^{\mathsf{T}} \boldsymbol{x}_i > \theta$ のときワーカー j はタスク i に対して 1 のラベルを，そうではないときに 0 のラベルを与える．このようにして，D 次元のベクトル空間上にワーカーとタスクを配置し，それらの位置関係を考慮してラベルを決定するため，ワーカーとタスクの相性を考えることができる．

2.3 イロレーティング

はじめに，イロレーティング [1], [2] が用いられている例やイロレーティングの性質について述べる．イロレーティングは，2 人のプレーヤー，または二つのチームが対戦し勝敗を決定するような対戦型の競技において，プレーヤー，チーム間の実力を相対的に表すために用いられる指標の一つである．イロレーティングは，もともとはチェスにおいて，プレーヤーの実力を相対評価するために導入された指標であるが，サッカー，ラグビー，テニスといったスポーツのランキング，オンラインゲームでのランキングなどにおいても，イロレーティングをベースとした指標が用いられている．競技プログラミングサイトにおいても，イロレーティングが参加者の実力指標として用いられていることが多い．このイロレーティングは，ユーザの実力を示す数値であり，数値が大きいほどそのユーザが強いことを示す．また，異なる 2 人のユーザに対して，レーティングの数値の差を計算すると，2 人のユーザのどちらがどれほど勝ちやすいか勝率を計算することができる．

イロレーティングの定義について述べる．レーティングが r_i のユーザ i と，レーティングが r_j のユーザ j が対戦することを想定する．このとき，ユーザ i がユーザ j に勝利する確率 p_{ij} は b, ξ をパラメータとして，

$$p_{ij} = \frac{1}{1 + b^{-(r_i - r_j)/\xi}} \tag{3}$$

で定義される．この式より，

$$\begin{aligned} \frac{p_{ij}}{p_{ji}} &= \frac{b^{r_i/\xi}}{b^{r_j/\xi}} \\ p_{ij} &= p_{ji}(b^{(r_i - r_j)/\xi}) \end{aligned}$$

が導かれる．つまり，ユーザ i のレーティングがユーザ j よりも ξ 高いと，ユーザ i がユーザ j に勝つ確率が b 倍高くなることを示している．このパラメータ b, ξ の値は，競技の主催者側が自由に設定することができる．

2.4 項目反応理論におけるロジスティックモデル

項目反応理論 [3] が競技プログラミングのタスクの難易度推定に関連する部分として，2 母数ロジスティックモデルを説明する．

2 母数ロジスティックモデルでは，問題の識別力と問題の困難度の二つのパラメータで問題の特性を表現するモデルである．ある問題 i を受験者が解くとき，問題 i の識別力を a_i，問題 i

の困難度を b_i，θ を受験者の能力を表す数値（大きいほど能力が高い），D を定数とすると，受験者 i が問題に正答できる確率 $p_i(\theta)$ は，

$$p_i(\theta) = \frac{1}{1+\exp(-Da_i(\theta - b_i))} \tag{4}$$

で表される．これを 2 母数ロジスティックモデルという．D は尺度因子と呼ばれる値で $D = 1.7$ とされることが多い．

問題の困難度 b_i は，その問題の難しさを表しており，値が大きいほどその問題が難しいことを表す．2 母数ロジスティックモデルにおいて，$\theta = b_i$ とおけば，$p_i(b_i) = 0.5$ となるので，この困難度の値は，問題に 50% の確率で正答できる能力レベルを数値化した母数と言い換えることができる．

問題の識別力 a_i は，能力値 θ の違いがどれほど正答確率に影響するかを決める母数である．この値は $\theta = b_i$ の時の傾きに比例する．傾きが大きいほど，能力値が変わると正答できる確率が大きく変化することを表す．

2.5 オフラインでのタスクの難易度の推定方法

競技プログラミングサイトにおけるタスクの難易度推定は，コンテスト終了後にコンテスト参加者の解答データを用いて難易度推定を行う手法が一般的である．

コンテスト形式と，コンテスト参加者に関する情報の扱いについて述べる．一般的な競技プログラミングのコンテストでは，様々な難易度のタスクが 3 から 7 問程度出題される．コンテスト参加者の解答データには，各タスクごとに正答，不正答，未提出などの解答状況データ，正答であれば正答した時刻が含まれている．また，コンテスト参加者自身のデータとしては主にユーザ名，ユーザの強さを表すレーティング，これまでのコンテスト参加回数などが含まれる．

現在，コンテスト終了後のタスクの難易度推定で用いられている手法は，AtCoder Problems[1] や Codeforces[2] によると，各タスクごとの解答状況データとユーザのレーティング情報を用いて，2.4 節で述べた 2 母数ロジスティックモデルを適用することによって推定を行っている．具体的な方法としては，まず，コンテスト参加者があるタスクについて，コンテスト終了時までに正答できていれば 1 のラベルを，そうでなければ 0 のラベルを目的変数としてもつ．そして，2 母数ロジスティックモデルにおいて，説明変数として各コンテスト参加者のレーティングを用いて，タスクの困難度 b_i とタスクの識別力 a_i を値を変化させながら，フィッティングを行う．このフィッティングによって得られたタスクの困難度 b_i をタスク i の難易度推定値として出力する．

3 提案手法

この章では，これまでコンテスト終了後にしかできなかったタスクの難易度推定を，コンテスト中にリアルタイムで推定する手法を提案する．

1 : https://kenkoooo.com/atcoder/

2 : "Codeforces: Problem Difficulties" https://codeforces.com/blog/entry/62865

3.1 タスクの難易度のリアルタイム推定について

はじめに，コンテストにおける参加者やタスクについてのデータの特性を述べる．実際のコンテストにおいては，コンテスト参加者の多くは難易度の低いタスクから順番に解答していくことが多い．また，レーティングが高い参加者の方が，難易度の高いタスクに正答できる時刻が早いことが多い．そのため，易しいタスクのデータは，コンテスト開始してから早い段階でさまざまなレーティングをもつ参加者の解答状況が集まる．一方，難易度の高いタスクでは，コンテスト開始してからある程度時間が経たなければ，解答状況が集まらないうえ，解答データが集まり始めた段階ではレーティングの高い参加者のデータが多く集まる．また，データ数自体も難易度の低いタスクよりも難易度の高いタスクの方が少ない．そのため，既存の手法でリアルタイム手法を行うと，高いレーティングを持つ参加者のデータのみを用いた難易度推定になってしまうため，難易度の高いタスクにおいては，コンテスト終了後に推定したタスクの難易度よりも大幅に高い難易度が出力されてしまうと考えられる．

本研究におけるタスクの難易度のリアルタイム推定では，コンテスト参加者のタスク正答状況と，タスク正答までにかかった時間，および参加者のレーティングに着目し，コンテスト開始の早い段階でできるだけコンテスト終了後のタスクの難易度推定で出力される値に近い値をコンテスト中に推定できるようにすることを考える．

提案手法は二つのパートから構成される．一つ目は，最尤推定パートであり，ここでは，コンテストのある時点での正答者のレーティングと，タスクに正答するのにかかった時間から，タスクの難易度を最尤推定する．二つ目は，予測パートで，ここでは，最尤推定パートで推定したタスクの難易度に基づいて，現時刻までに正答できていない参加者のうち，コンテスト終了時までに正答できるレーティングはいくつかを予測する．

3.2 最尤推定パート

最尤推定パートについて説明する．

はじめに，最尤推定パートのモデルについて説明する．i 人目のコンテストの参加者は，タスクに時間 T ごとに解答を提出し，提出ごとに確率 p_i で正答，確率 $1-p_i$ で不正答の判定を受けるものとする．p_i の具体的な式については後述する．正答の判定を受けた時点でこのタスクは終了したものとみなし，不正答の判定を受けた場合は，時間 T ごとに正答の判定を受けるまで回答を提出し続けるものとする．このモデルを図で表したものを図 1 に示す．

次に，前述のモデルを定式化する．タスクの難易度を R_{mle}，タスクに正答したコンテスト参加者のうち，i 人目のタスク正答者を u_i と表すことにする．u_i のレーティングを r_i とする．u_i がタスクに正答できる確率 p_i は，2.3 節で述べたイロレーティングの定義式 (3) より，$b(>1), \xi(>0)$ をパラメータとして，

$$p_i = \frac{1}{1+b^{-(r_i - R_{mle})/\xi}}$$

と表せる．タスクに 1 回解答を提出するのにかかる時間を T と

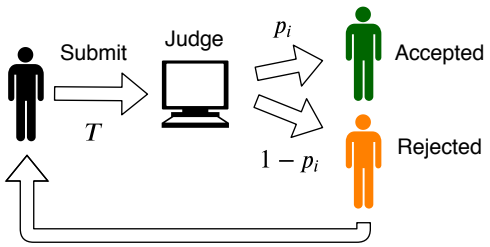

図 1　最尤推定パートのモデル

おくと，u_i が正答するのにかかる時間の期待値 e_i は，提出回数の期待値が p_i の逆数で表せることから，

$$e_i = T \times \frac{1}{p_i} \tag{5}$$

と表せる．ここで，コンテスト参加者があるタスクに自分の解答を繰り返し提出し，正答であるという判定を得るまでにかかる時間が指数分布に従うと仮定する．指数分布とは，ある期間に平均で λ 回起こる現象が，次に起こるまでの期間 X が従う分布のことである．指数分布を適用する理由としては，コンテスト参加者がタスクと擬似的に対戦していることを想定し，コンテスト参加者が初めてタスクに正答するという事象が起こるまでにかかる時間を表すのに適していると考えられるからである．確率変数 X が母数 λ の指数分布に従うとき，$X = x$ となる確率密度関数 $Pr(X = x)$ は，

$$Pr(X = x) = \begin{cases} \lambda \exp(-\lambda x) & (x \geqq 0) \\ 0 & (x < 0) \end{cases}$$

と表される．また，この指数分布の平均は $1/\lambda$ である．(5) 式と指数分布の平均から，

$$\frac{1}{\lambda} = e_i$$
$$\lambda = \frac{p_i}{T}$$

と表すことができる．前のタスクに正答した時刻から，現在のタスクに正答するまでに u_i がかかった時間を t_i とする．タスクに N 人正答しているとき，尤度関数 $L(R_{mle})$ を指数分布の確率密度関数を用いて，

$$L(R_{mle}) = \prod_{i=1}^{N} \frac{p_i}{T} \exp\left(-\frac{p_i}{T} t_i\right)$$

と定める．対数をとると，

$$\begin{aligned} &\log L(R_{mle}) \\ &= \sum_{i=1}^{N} \log \frac{p_i}{T} \exp\left(-\frac{p_i}{T} t_i\right) \\ &= -N \log T + \\ &\quad \sum_{i=1}^{N} \left(-\log\left(1 + b^{-(r_i - R_{mle})/\xi}\right) - \cdot \frac{t_i/T}{1 + b^{-(r_i - R_{mle})/\xi}} \right) \end{aligned} \tag{6}$$

となる．T について偏微分すると，

$$\frac{\partial \log L(R_{mle})}{\partial T} = -\frac{N}{T} + \frac{1}{T^2} \sum_{i=1}^{N} \frac{t_i}{1 + b^{-(r_i - R_{mle})/\xi}}$$

と表せる．(6) 式の対数尤度関数を最大化する R_{mle} を求めることにする．この R_{mle} の値は，タスクの難易度のとりうる値の最小値を r_{min}，最大値を r_{max} とすると，区間 $[r_{min}, r_{max}]$ に含まれる整数を全探索することによって近似的に求める．R_{mle} の値を固定したあと，T の値を，偏微分の式が 0 となる条件から，

$$T = \frac{1}{N} \sum_{i=1}^{N} \frac{t_i}{1 + b^{-(r_i - R_{mle})/\xi}}$$

で求め，(6) 式の対数尤度関数が最大となる (R_{mle}, T) のペアを求め，この時の R_{mle} の値を，最尤推定パートで推定されるタスクの難易度とする．

3.3　予測パート

次に，予測パートについて説明する．このパートでは，タスクに正答していないコンテスト参加者が，現在の時刻よりも後に正答できる状況を考慮し，先ほど求めた R_{mle} の値を指数分布の累積確率分布を用いて予測することを考える．確率変数 X が母数 λ （期待値 $1/\lambda$）の指数分布に従うとき，あるコンテスト参加者が時間 $X = x$ までにタスクに正答する確率 $Pr(X \leqq x)$ は，

$$Pr(X \leqq x) = \begin{cases} 1 - \exp(-\lambda x) & (x \geqq 0) \\ 0 & (x < 0) \end{cases} \tag{7}$$

となる．求めたいタスクの難易度を D_{cum} とおくと，最尤推定で求めたタスクの難易度推定値 R_{mle} の値を用いて，まだタスクに正答できていないコンテスト参加者が 1 回の提出で正答できる確率 P は，イロレーティングの定義式 (3) より，

$$P(D_{cum}) = \frac{1}{1 + b^{-(D_{cum} - R_{mle})/\xi}}$$

と表せる．また，最尤推定パートで定義した，タスクに 1 回解答を提出するのにかかる時間 T を用いると，タスクに正答できていないコンテスト参加者がタスクに正答するのにかかる時間の期待値は，$T/P(D_{cum})$ となり，これより，

$$\lambda = \frac{P(D_{cum})}{T}$$

である．コンテストの残り制限時間を z，残っているタスクの数を q とする．q の値は，全てのコンテスト参加者がタスク番号の若い順に問題を解いていくものと仮定して決定する．例えば，コンテストで 6 問のタスクが出題されており，タスクのラベルが A から F まであり，タスク D の難易度を推定したい時は残っているタスクは D, E, F であるから $q = 3$ である．各タスクに対してコンテストの残り制限時間を均等に配分したと仮定すれば，各タスクにそれぞれ z/q だけ時間をかけることができる．これらを用いると，まだタスクに正答できていないコンテスト参加者が，時間 z/q 以内に正答できる確率 $Pr'(X \leqq z/q)$ は，

$$\begin{aligned} Pr'(X \leqq z/q) &= 1 - \exp\left(-\frac{P}{T} \cdot \frac{z}{q}\right) \\ &= 1 - \exp\left(-P \cdot \frac{z/q}{T}\right) \end{aligned} \tag{8}$$

表 1　レーティングと各レーティング帯の存在人数

レーティング	順位下限（目安）	上位%
3000〜	30 位程度	0.04 %
2600〜2999	180 位程度	0.25 %
2400〜2599	460 位程度	0.65 %
2300〜2399	680〜700 位程度	1 %
2100〜2299	2000 位程度	2.8 %
1900〜2099	4500〜4600 位	6.5 %
1600〜1899	13000〜13500 位	19 %
1400〜1599	30000 位程度	42 %
1200〜1399	58600 位程度	82 %
〜1199	71000 位程度	100 %

と表すことができる．D_{cum} ここで，残り時間 z が 0 に近づくと，D_{cum} は大きな値に発散してしまうため，残り時間で提出できる回数 y を，

$$y = \max\left(1, \frac{z/q}{T}\right)$$

と定義し，残り時間 z が減少すると D_{cum} が収束するように修正する．これを用いて，(8) 式を修正し，まだタスクに正答できていないコンテスト参加者が，時間 z/q 以内に正答できる確率 $Pr(X \leqq z/q)$ を改めて定義すると，

$$\begin{aligned} & Pr(X \leqq z/q) \\ &= 1 - \exp\left(-P \cdot y\right) \\ &= 1 - \exp\left(-\frac{1}{1 + b^{-(D_{cum} - R_{mle})/\xi}} \max\left(1, \frac{z/q}{T}\right)\right) \end{aligned} \quad (9)$$

となる．この (9) 式の $Pr(X \leqq z/q)$ の値が 0.5 となる D_{cum} を二分探索で求め，D_{cum} の値をそのタスクのリアルタイムでの難易度推定値として出力する．このリアルタイムでの出力値と，コンテスト終了後にタスクの難易度推定を行って得られる値との差が，コンテスト開始からできるだけ早い時刻で小さくなることが，リアルタイムでの難易度推定値の望ましい出力値である．

4　実　　験

4.1　実験で用いたデータ

今回の実験で用いたデータについて説明する．

はじめに，コンテスト参加者のレーティングデータの扱いについて述べる．レーティングのデータは各コンテスト参加時点での値ではなく，2020 年 2 月 12 日時点でのレーティングの値を用いた．イロレーティングの定義式 (3) において，パラメータの値は，Codeforces が用いている $\xi = 400, b = 10$ に固定した．

Codeforces におけるユーザのレーティング分布とタスクの難易度について述べる．2020 年 1 月 30 日から直近 6ヶ月以内のコンテストに 1 度でも参加したユーザのレーティング分布は図 2 のようになっている．また，レーティングの値と，アクティブユーザの各レーティング帯に存在する人数と比率は表 1 のようになっている． Codeforces では，登録時はレーティング 1500 から開始し，1 回目のコンテストに参加すると初期値 1500 からレーティングが変動する仕組みになっている．また，タスクの難易度は，レーティングの値と紐づいている．

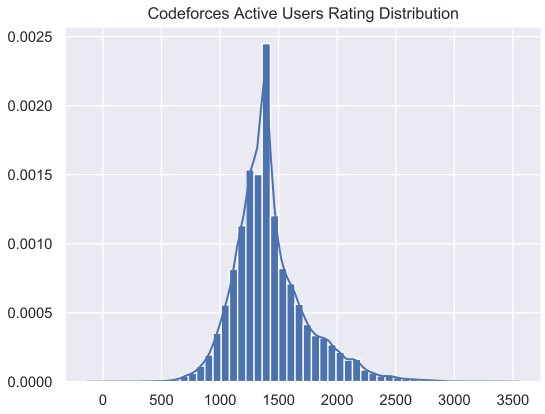

図 2　アクティブユーザのレーティング分布

今回の実験で用いたコンテストデータについて述べる．今回の実験では，Codeforces のコンテストのコンテストデータを用いた．正解の値が難易度 1800 以下のタスクについては，レーティング 1600 未満の参加者をレーティング変動対象とした Div.3 のコンテスト，またはレーティング 1900 未満かレーティング 2100 未満をレーティング変動対象とした Div.2 のコンテストのデータを用い，難易度 1900 以上 2200 未満のタスクについては十分な正答者数を確保するためレーティング 1900 以上がレーティング変動対象となる Div.1 のコンテストのデータを用いている．タスクの難易度推定には，Codeforces 側が難易度の推定値を 1000 以上 2100 以下としているタスクのデータを用いた．難易度が 1000 未満のものは，コンテストのほぼ全ての参加者がコンテストの早い時刻に正答しているため，今回の実験からは除外する．また，Codeforces 側が推定難易度 2200 超過としている問題については，正答データが少ないタスクが多く，手法自体を適用することは可能であるが，予測が不安定なものが多く存在するため除外する．以下，便宜上難易度 1000 以上 1500 未満を低難易度，1500 以上 1900 未満を中難易度，1900 以上 2100 以下を高難易度と呼ぶことにする．

評価に用いたコンテストのデータの前処理方法について述べる．元の順位表データのうち，リアルタイムでコンテストに参加した者のデータ（レーティング変動対象者，およびレーティング変動対象外の両方を含む）のみを用いることとし，コンテスト終了後以降に練習として行えるバーチャルコンテストの記録など，リアルタイムでコンテストに参加していない人のデータは前処理で除外した．また，2020 年 2 月 12 日以前にコンテストのルール違反などによって凍結されたユーザのデータも今回のタスクの難易度推定に用いるデータから除外した．

4.2　ロジスティックモデルによる推定の手順

ロジスティックモデルを用いてタスクの難易度をリアルタイムで推定する手順を以下に示す．時刻 t_c 分におけるタスクの

難易度の推定値を求める手順は以下の通りである．ここでは，i 人目のコンテスト参加者を u_i と表すことにする．

（1） コンテストの順位表のデータを取得する．

（2） u_i のレーティングを r_i とし，u_i が時刻 t_c 分までにタスクに正答できていれば $y_i = 1$，そうでなければ $y_i = 0$ とする．

（3） u_i がタスクに正答できる確率 p_i を

$$p_i = \frac{1}{1 + \exp\left(-(\alpha_0 + \alpha_1 r_i)\right)}$$

で定義し，尤度関数 $L(\alpha_0, \alpha_1)$ を

$$L(\alpha_0, \alpha_1) = \prod_i p_i^{y_i} (1 - p_i)^{1 - y_i}$$

とする．この尤度関数を最大化する α_0, α_1 の値を求める．

（4） 2.4 節の 2 母数ロジスティックモデルの式 (4) より，a をタスクの識別力，b をタスクの難易度，θ を能力値, D をパラメータとすると

$$p(\theta) = \frac{1}{1 + \exp(-Da(\theta - b))}$$

と表せる．前の手順で得られたパラメータ α_0, α_1 を用いると

$$\alpha_0 = -Dab$$

$$\alpha_1 = Da$$

と書けるので，時刻 t_c におけるタスクの難易度の推定値 E は $E = -\alpha_0/\alpha_1$ である．

（5） E の値を時刻 t_c におけるタスクの難易度の推定値として出力する．

この一連の手順を $t_c = 10$ から (コンテスト終了時刻) 分まで 1 分おきに行う．

4.3 最尤推定パートのみを用いた推定の手順

3.2 節の最尤推定パートを用いて，タスクの難易度をリアルタイムで推定する手順を以下に示す．時刻 t_c 分におけるタスクの難易度の推定値を求める手順は以下の通りである．

（1） コンテストの順位表のデータを取得する．

（2） コンテスト開始時刻から時刻 t_c 分までにタスクに正答したユーザを抽出する．

（3） コンテスト開始時刻から時刻 t_c 分までのタスクの正答者が 10 人未満なら，最高難易度として出力する．

（4） そうでなければ，3.2 節の (6) 式を最大化するようにタスクの難易度 R_{mle} とタスクに 1 回提出するのにかかる時間 T を求める．

（5） R_{mle} の値を時刻 t_c 分におけるタスクの難易度の推定値として出力する．

この一連の手順を $t_c = 10$ から (コンテスト終了時刻) 分まで 1 分おきに行う．

4.4 最尤推定パートと予測パートによる推定の手順

3.2 節の最尤推定パートと，3.3 節の予測パートを用いて，タスクの難易度をリアルタイムで推定する手順を以下に示す．時刻 t_c 分におけるタスクの難易度の推定値を求める手順は以下の通りである．

（1） コンテストの順位表のデータを取得する．

（2） コンテスト開始時刻から時刻 t_c 分までにタスクに正答したユーザを抽出する．

（3） コンテスト開始時刻から時刻 t_c 分までのタスクの正答者が 10 人未満なら，最高難易度として出力する．

（4） そうでなければ，3.2 節の (6) 式を最大化するようにタスクの仮の難易度 R_{mle} を求める．

（5） 3.3 節の (8) 式において，コンテスト残り時間 z と残りの問題数 q を求めて，$Pr(X \leqq z/q) = 0.5$ となるときの D_{cum} の値を求める．

（6） D_{cum} の値を時刻 t_c 分におけるタスクの難易度の推定値として出力する．

この一連の手順を $t_c = 10$ から (コンテスト終了時刻) 分まで 1 分おきに行う．

4.5 評価方法

手法間での性能の評価の方法について述べる．定量的な評価手法については，コンテスト開始時刻から 1/4 経過時点，1/2 経過時点，コンテスト終了時までの 1 分ごとの予測の平均絶対誤差を設ける．

平均絶対誤差による評価の方法を述べる．各手法において，時刻 $t_c = 10$ から (コンテスト終了時刻) 分まで 1 分おきにタスクの難易度の推定を行い，各時刻でのタスクの難易度の推定値を記録する．そして，各時刻において得られた推定値と，コンテスト終了後に Codeforces 側が設定した難易度値の差の絶対値を計算し，この値の平均値を求めることとする．平均絶対誤差が小さいほど性能が良い．

4.6 実験結果

提案手法と比較手法を用いて，様々なコンテストにおいてタスクの難易度を推定した．

はじめに，定性的な傾向について説明する．図 3，図 4，図 5 は，各時刻において手法ごとの難易度推定値を，横軸をコンテスト開始からの経過時間（単位:分），縦軸を推定難易度とした折れ線グラフにプロットしたものである．青い線（ラベル: MLE）が 4.3 節で述べた最尤推定パートのみを用いた推定，橙色の線（ラベル: MLE + Cumulative）が 4.4 節で述べた最尤推定パートと予測パートを用いた推定，緑色の線（ラベル: logistic）が 4.2 節で述べたロジスティックモデルによる推定，赤色の線（ラベル: Correct）が Codeforces 側が設定したそのタスクの難易度推定値である．各手法において，推定値の折れ線と赤色の正解値の直線の値の差が小さい方が推定値の精度が良く，また早い時刻に正解値に近づく方が性能が良い．どの難易度のタスクについても，ロジスティックモデルでは高い値から徐々に難易度の予測値が下降していく傾向がみられたのに対し，提案手法のモデルでは，低い値の予測値から徐々に上昇し

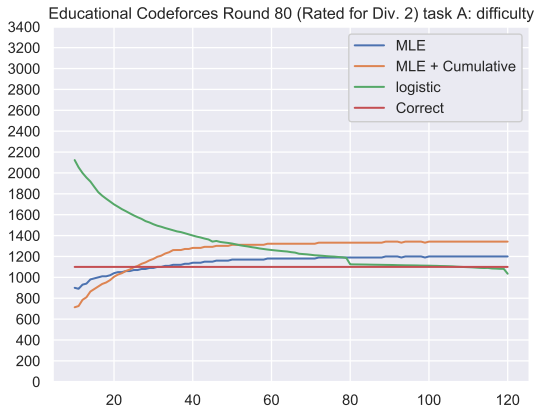

図 3　推定値の推移 (低難易度)

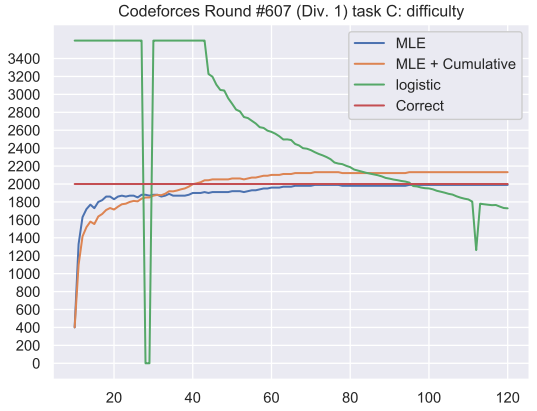

図 4　推定値の推移 (中難易度)

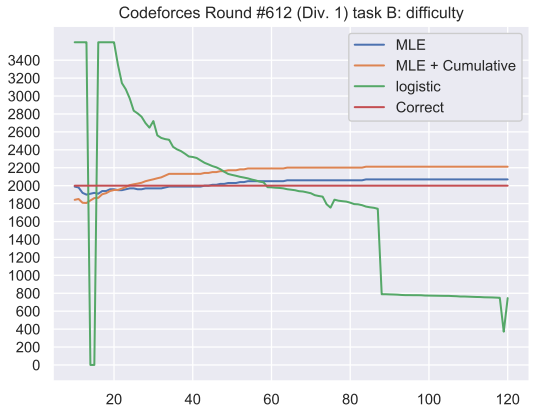

図 5　推定値の推移 (高難易度)

ていく傾向がみられた．提案手法で，予測値が上昇していくのは，初期の推定では，タスクに解答を 1 回提出するのにかかる時間 T の値が小さく，タスクに対して解答を提出する周期が短くなることと，後半では，残り時間が短くなりタスクに解答を提出できる回数が少なくなるからだと考えられる．中難易度から高難易度の問題については，提案手法の方がコンテスト開始の早い時刻において正解の値により近い値を出力しており，比較手法よりも良い結果が得られている．低難易度の問題については，コンテスト開始からまもない時刻においては，提案手法の方が正解の値に近いが，後半になるとロジスティックモデルの方が正解の値に近づくことが多かった．

表 2　リアルタイム推定での平均絶対誤差 (低難易度)

難易度	タスク数	手法	1/4	1/2	ALL
1000	22	最尤推定	174	**218**	254
		最尤推定+予測	**130**	219	329
		ロジスティック	584	389	**218**
1100	9	最尤推定	**94**	**126**	**162**
		最尤推定+予測	123	174	259
		ロジスティック	707	489	294
1200	12	最尤推定	**142**	**144**	**156**
		最尤推定+予測	225	220	256
		ロジスティック	796	538	325
1300	19	最尤推定	**264**	**172**	**135**
		最尤推定+予測	348	224	206
		ロジスティック	928	611	367
1400	11	最尤推定	**285**	**173**	**109**
		最尤推定+予測	387	209	174
		ロジスティック	1118	713	414

次に，絶対平均誤差一部抜粋したものを以下の表 2，表 3，表 4 に示す．表中の値は，実際の値を小数第 1 位で四捨五入している．表中の 1/4 はコンテスト開始から最初の四分の一の時間までの予測値の平均絶対誤差，1/2 はコンテストの前半における予測値の平均絶対誤差，ALL はコンテスト開始から終了までの予測値の平均絶対誤差である．また，タスク数は今回の推定に用いたタスクデータ数である．各タスクにおいて最も精度が高かったものを太字で表している．比較手法のロジスティックモデルと提案手法を比較すると，最尤推定パートのみを行った場合や，最尤推定パートと予測パートを行ったものが精度が高かった．予測パートを行った場合の方が精度が悪くなった原因としては，まだ正答できていない人の人数などを考慮できていないことや，最尤推定パートの予測値を前提と予測を行っているため，こちらの精度が悪いと予測パートでも精度が悪くなってしまうことが考えられる．また，一番精度が良かった手法でも，平均絶対誤差が 300 を超えている場合も多かった．これは，ロジスティックモデルとは異なり，最尤推定をベースとした予測のため，レーティングと正答率の関係をあまり考慮していないことが原因であると考えられる．

5　結　　論

本研究では，競技プログラミングサイトのコンテストにおいて，リアルタイムでタスクの難易度を推定する手法を提案した．提案手法では，はじめに，コンテスト参加者のある時刻までの正答状況と，正答するまでにかかった時間の情報を用いて，最尤推定を行った．続いて，まだタスクに正答できていない参加者のうち，コンテスト終了時までに 50%の確率で正答できるレーティングの値はいくつかを推定する方法を考案した．実際のコンテストのデータを用いて，実験を行い評価をしたところ，

表 3　リアルタイム推定での平均絶対誤差 (中難易度)

難易度	タスク数	手法	1/4	1/2	ALL
1500	18	最尤推定	**429**	**307**	234
		最尤推定+予測	526	308	**213**
		ロジスティック	1262	799	457
1600	13	最尤推定	**640**	468	310
		最尤推定+予測	717	**438**	**254**
		ロジスティック	1542	998	571
1700	14	最尤推定	**809**	566	408
		最尤推定+予測	848	**514**	**310**
		ロジスティック	1693	1125	628
1800	12	最尤推定	**929**	702	543
		最尤推定+予測	1011	**672**	**456**
		ロジスティック	1754	1421	846

表 4　リアルタイム推定での平均絶対誤差 (高難易度)

難易度	タスク数	手法	1/4	1/2	ALL
1900	7	最尤推定	**203**	**140**	**109**
		最尤推定+予測	255	165	161
		ロジスティック	1364	984	1085
2000	6	最尤推定	**186**	**147**	**114**
		最尤推定+予測	296	200	181
		ロジスティック	1444	960	704
2100	3	最尤推定	**487**	**298**	**181**
		最尤推定+予測	656	346	194
		ロジスティック	1444	1134	694

提案手法における予測値の方が正解の値に近づくのが早く，提案手法の有効性が確認できた．

本研究の今後の課題について述べる．

まず，提案手法では様々なパラメータが現れたが，この決め方を改良することができると考えている．タスクに正答できていない参加者の存在を加味する際に，コンテストの残り時間を正答できていないタスクに均等に配分していた．しかし，実際のコンテストでは，易しい問題から順番に解く参加者が多いことや，難しいタスクは解けないので時間をほとんどかけないということ，コンテスト終盤では 1 問に集中して取り組むことなど，コンテスト参加者にはある程度決まった取り組み方があり，これらの点を考慮できていない．そのため，他のタスクの推定難易度を比較して，残り時間の配分に重み付けをすることを考えるとよいと思われる．また，タスクに 1 回解答を提出するのにかかる時間 T が一定という過程についても再検討する必要があると考えている．

次に，今回の提案手法では，各タスクごとにタスクの難易度を推定したが，コンテストでは一度に複数のタスクが出題されるので，ほかのタスクの正答状況をタスクの難易度推定に使えると考えられる．

また，参加者全体のうちまだ解けていない人数の割合や，残っている参加者のレーティング分布などを考慮すると，よりリアルタイムでタスクの難易度を推定する精度が上がるのではないかと考える．

最後に，実験ではいくつかのコンテストを抽出して実験を行なったが，もっと多くのコンテストデータを用いて実験を行う必要があると考えている．

6 謝　　辞

本研究は，JST CREST (JPMJCR16E3)，JSPS 科研費 18H03245 の支援を受けたものである．

文　　献

[1] Arpad E. Elo. *The Rating of Chess Players, Past and Present*. Arco Publishing Company, 1978.

[2] 岩野和生, 中村英史, 清水咲里. 『レイティング・ランキングの数理　ー No.1 は誰か？』. 共立出版株式会社, 2016.

[3] 豊田秀樹.『項目反応理論 [入門編]　ーテストと測定の科学ー』. 株式会社朝倉書店, 2002.

[4] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[5] Nicholas I Sapankevych and Ravi Sankar. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine*, Vol. 4, No. 2, pp. 24–38, 2009.

[6] Jacob Whitehill, Ting fan Wu, Jacob Bergsma, Javier R. Movellan, and Paul L. Ruvolo. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pp. 2035–2043. Curran Associates, Inc., 2009.

[7] Peter Welinder, Steve Branson, Pietro Perona, and Serge J. Belongie. The multidimensional wisdom of crowds. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pp. 2424–2432. Curran Associates, Inc., 2010.