

コンピュータ大貧民におけるレーティング手法の最適化に関する研究

藤村 光希¹ 大久保 誠也² 若月 光夫¹ 西野 哲朗¹

概要：本研究の目的は、コンピュータ大貧民におけるレーティングシステムの確立である。そのために、五ヶ谷らが提案した手法に、ゲーム固有の性質や統計情報を組み込んだ手法を提案する。提案手法は4つある。1つ目は、複数人対戦として解釈しレートを出算する手法、2つ目は、コンピュータ大貧民に適したランダム性を追加する手法、3つ目は、ある一定の得点差を引き分けと考慮する手法、4つ目は、プレイヤーの得点情報からスキルの低いプレイヤーに負の見積もりを与える手法である。提案手法の有効性を検証するために、計算機実験を行った。実験は、4つの提案手法を段階的に追加し、追加前と追加後の手法を比較した。具体的には、レートと、標準偏差から求められるプレイヤーの予測勝率と、実際の対戦結果から得られるプレイヤーの実際勝率との平均二乗誤差を比較した。そして、平均二乗誤差の大きさからレートの精度が向上したか否かを調べた。実験の結果、コンピュータ大貧民においては、複数人対戦として解釈すること、ゲームのランダム性を小さいものとする、一定の得点差を引き分けと見なすこと、一定の得点以下のプレイヤーをスキルが低いと見積もることにより、先行研究よりもレートの精度が向上することが確認できた。これらの実験結果から、各種提案手法は有効であるといえる。

キーワード：コンピュータ大貧民, レーティング, ベイズ統計学, TrueSkill

1. はじめに

プレイヤーが3名以上で、各プレイヤーの持つ情報に違いがあるゲームを、多人数不完全情報ゲームという。多人数不完全情報ゲームは、予測しなければならぬ手が非常に多く、強いプレイヤーAIの実現が困難であるとされていた。しかし、2019年に、Microsoft Research Asiaが開発した麻雀AIであるSuphxが、オンライン麻雀サービスで、全プレイヤーの上位0.0054%にあたる10段を達成した[1]。このように多人数不完全情報ゲームの分野での研究が進んできている。

多人数不完全情報ゲームの1つとしてトランプゲームの大貧民がある。大貧民AIの強さを競う大会であるUECコンピュータ大貧民大会(UECda)が、毎年、電気通信大学で開催されている[2]。しかしながら、開催年度をまたいだ強さの評価はなされておらず、強さを定量的に評価する方法が求められていた。ゲームにおけるプレイヤーの強さを定量的に評価したものがレートである。五ヶ谷らは、複数人対戦で行われるコンピュータ大貧民の結果を1対1の結果として解釈し、既知のレーティングシステムを適用する手法を提案した。そして、大貧民AIのレートを求めることで

プレイヤーの強さを定量的に評価した[3]。その結果、既知のレーティングシステムの1つであるTrueSkill[4]を使用したときに、最も精度が良くなるという結論を得た。しかしながら、多人数ゲームを2人ゲームとして解釈している点、大貧民というゲームの特徴を捉えていない点など、検討点や課題が残されている。

本研究の目的は、コンピュータ大貧民に特化した、高い精度を持つレーティング手法の確立である。具体的には、五ヶ谷らの手法に対し、コンピュータ大貧民の性質や、強いAIの特徴等の知見を組み込んだ4つの手法を提案する。そして、計算機実験により、有効性の検証を行う。具体的には、レートの差から求められる予測勝率と、実際の対戦データから得られる実際勝率との平均二乗誤差を比較する。

2. コンピュータ大貧民

コンピュータ大貧民では、5体の大貧民AIによって対戦が行われる。ここで、コンピュータ大貧民における1回のゲームは、UECルールに則り、以下のように行われる[5]。

- (1) 各スートのA~Kの52枚にジョーカーを1枚加えた計53枚のカードをプレイヤーに均等に配布する。
- (2) ◇の3を持っているプレイヤーから手番を開始する。

¹ 電気通信大学大学院 情報理工学研究所

² 静岡県立大学 経営情報学部

- (3) 以下を、プレイヤーが1人になるまで繰り返す。
- (a) 各プレイヤーはカードの強さにしたがって、場にカードを提出する。もしくはパスを行う。
 - (b) 手札が無くなったプレイヤーは、ゲームから抜ける。また、すべてのプレイヤーがパスすると、場のカードは空になり、最後にカードを出したプレイヤーの手番となる。
- (4) 先に抜けたプレイヤーから大富豪、富豪、平民、貧民、大貧民となる。また、それぞれ5, 4, 3, 2, 1の得点が与えられる。

2回目以降のゲームでは、(1)と(2)の間に以下のステップが加わる。

- (1') 1つ前のゲームで大富豪になったプレイヤーは大貧民とカードを2枚交換する。その際、大富豪は自分の配られた手札の中から任意の2枚を選択し、大貧民に渡す。大貧民は自分の配られた手札の中からカードの強さが1番強いカードと2番目に強いカードを選択し、大富豪に渡す。富豪と貧民の間にも同様の手順で手札を1枚交換する。平民は何もしない。

コンピュータ大貧民では、上記の1回のゲームを複数回繰り返し、累積得点を競う。ゲームを一定数繰り返すことを1セットと呼ぶ。コンピュータ大貧民は、多人数不完全情報ゲームであるため、1回のゲームはランダム性が強い。

3. レーティング

対戦ゲームにおけるプレイヤーの強さは、試合の結果から直接判断することが難しい。そこで開発されたのが、レートである。レートは、他プレイヤーのレートと比較することで、プレイヤーの強弱を判断できる。ゲームの結果からレートを求めるまでの、一連の処理を実施するアルゴリズムやプログラムをレーティング手法と呼ぶ。これまでに、Elo[6]など、多くのレーティングシステムが提案されてきている。

コンピュータ大貧民におけるレーティングについても、いくつかの研究が行われている[3][7]。Eloなどの多くのレーティングシステムは2人ゲームでの使用を前提としている。そこで五ヶ谷らは、5人対戦でゲームが行われるコンピュータ大貧民を2人ゲームの結果として解釈し、2人用ゲームのレーティングシステムを適用する手法を提案した。また、計算機実験により、その有効性を検証した。実験内で使用されたレーティングシステムは、Elo, Glicko[8], Glicko2[9], TrueSkillの4つである。その結果、TrueSkillを用いたときの精度がもっともよくなった。

4. TrueSkill

TrueSkillは、Microsoft Researchによって開発されたレーティングシステムである。TrueSkillはEloと比較したとき、

「複数人対戦ゲームの結果に対して適用できる」、「レートが収束するため、発散が起こらない」、「引き分けを扱える」という性質を持つ。

TrueSkillは、プレイヤーの強さを「スキル」と表記する。プレイヤーを i 、スキルを s_i 、スキルの平均を μ_i 、スキルの標準偏差を σ_i としたとき、スキルは、式1のように正規分布にしたがうと仮定される。

$$P(s_i) = N(s_i; \mu_i, \sigma_i^2) \quad (1)$$

式1の表記は、確率変数 s_i の分布であることを示す。ここで、スキル s_i が与える試合の結果 r の確率モデルを $P(r|s_i)$ とする。このとき、ベイズの定理を適用すると、試合が行われた後のスキルは、以下の式で表現される。

$$P(s_i|r) = \frac{P(r|s_i)P(s_i)}{P(r)} \quad (2)$$

ベイズの定理にしたがって、式2の $P(s_i)$ を事前分布、 $P(r|s_i)$ を尤度、 $P(s_i|r)$ を事後分布と呼ぶ。このとき、事前分布と尤度は共に正規分布にしたがうと仮定される。これらは自然共役分布の関係にあるため、計算結果である事後分布も正規分布にしたがう[10]。事後分布 $P(s_i|r) = N(\mu_{update}, \sigma_{update}^2)$ としたとき、このときの μ_{update} は、更新されたレートを表す。このとき、尤度 $P(r|s_i)$ は複数の確率変数による同時確率として表現される。この複数の確率変数では、ゲームの性質などが表現されており、ゲームの性質を確率分布として、レーティング計算に複数取り込めることを示している。

TrueSkillには、レーティングの環境を設定するパラメータを指定できる。以下にパラメータを示す。

初期のスキルの平均: μ_{init}

初めて試合に参加したプレイヤーに与えられる初期のスキルの平均。

初期のスキルの標準偏差: σ_{init}

初めて試合に参加したプレイヤーに与えられる初期のスキルの標準偏差。推奨値は $\sigma_{init} = \frac{\mu_{init}}{3}$ である。

ゲームのランダム性を導入するパラメータ: β

スキルに、対象のゲームのランダム性を導入するパラメータ。 β は対象のゲームにより異なるため、調整が必要である。この β を追加したスキルをパフォーマンスと呼ぶ。

レートの更新が滞るのを制限するパラメータ: η

レートが完全に収束してしまうことを防ぐパラメータ。推奨値は、 $\frac{\sigma}{100}$ である。

2人のプレイヤー間で引き分けが起こる確率: $draw_prob$

引き分けが発生する確率。対象ゲームにより異なるため、調整が必要である。

TrueSkillを用いてレートを求める手順を示す。まず、使用するデータ構造は以下のとおりである。

順位リスト: r

上位から順にプレイヤー番号を並べたリスト.

全プレイヤーのスキルリスト :all_player_list

試合に参加した全プレイヤー名と、プレイヤーごとのスキル μ, σ を格納するリスト. 試合が行われるごとに、試合に参加したプレイヤーのスキルは、更新されたスキル $\mu_{update}, \sigma_{update}$ で上書きされる.

現在のプレイヤーのスキルリスト :player_list

現在の 1 試合に参加したプレイヤーのスキルを all_player_list から抽出したリスト.

TrueSkill では、これらのデータ構造に格納されたデータを、以下のように更新する.

- (1) 環境パラメータ $\mu_{init}, \sigma_{init}, \beta, \eta, draw_prob$ を設定.
- (2) 以下を対象ゲームの試合数分だけ繰り返す.
 - (a) 1 試合の結果を整形・補正した順位リスト r を作成.
 - (b) 試合に参加したプレイヤー名と、スキル μ, σ を all_player_list から抽出し、player_list に格納する. all_player_list に登録されていないプレイヤーは、初期のスキル $\mu_{init}, \sigma_{init}$ を player_list に格納する.
 - (c) (1) で指定したパラメータ、順位リスト r , player_list を TrueSkill システムに入力する.
 - (d) TrueSkill の内部でレートが更新が行われる. TrueSkill システムから更新されたスキル $\mu_{update}, \sigma_{update}$ を受け取り、player_list に上書きする.
 - (e) player_list に格納されている更新されたスキルを all_player_list に登録する.
- (3) all_player_list に登録されている全プレイヤー名と、スキル μ, σ を出力する.

上記ステップの (2).(d) では、TrueSkill の内部システムでのレートが更新が行われる. 内部では、確率伝搬法を使用することで、式 2 を効率的に求めている.

Microsoft Research は TrueSkill の後続である TrueSkill2 を開発した [11]. TrueSkill2 は、各種統計情報を利用することで、レート精度を向上させている. 統計情報を利用しているため、対象としていたシューティングゲームに特化したレーティングシステムとなっている.

5. 提案手法

5.1 提案手法 1: 複数人対戦の結果としてレートを算出する手法

五ヶ谷らの提案手法では、コンピュータ大貧民の試合結果を 2 人ゲームの対戦と解釈し、既存のレーティングシステムを使用することでレートを算出した. しかしながら、TrueSkill は複数人対戦の結果を適用し、レートを算出でき

る. そこで、2 人ゲームとしての解釈を挟むことなく、直接レートを求める手法を提案する. 提案手法 1 の手順を以下に示す.

- ステップ 1: 大貧民 AI5 体による 10000 ゲームからなる 1 セットの累積得点を取得.
- ステップ 2: 五ヶ谷らの提案手法と同様の環境パラメータ設定する.
- ステップ 3: ステップ 1 で得られた累積得点に対して、点数の高いものから 1~5 の順位を割り当て、試合の順位リストを作成する.
- ステップ 4: 順位リスト、プレイヤーの現在のスキル、環境パラメータを TrueSkill に入力し、レートを算出する.
- ステップ 5: 行ったセット数分だけ、上記ステップを繰り返す.

5.2 提案手法 2: 対象のゲームに適したパラメータ β を考慮してレートを算出する手法

TrueSkill には、将棋やポーカーのような対象とするゲームにおけるランダム性を導入できるパラメータ β がある. 五ヶ谷らの提案手法では、 $\beta = 200$ の結果のみでしかレートを算出していない. コンピュータ大貧民の 10000 ゲームを 1 セットとしたときのゲームのランダム性を表現するためには、適した β を検討する必要がある. コンピュータ大貧民は 10000 ゲームを 1 セットとしてその累積得点を競うため、大数の法則により、ランダム性の影響を小さくできる. そのため、コンピュータ大貧民の対戦結果に使用するゲームのランダム性を表現するパラメータ β は小さい値が適していると考えられる. そこで、提案手法 2 として、ゲームのランダム性を小さく設定することを提案する. 提案手法 2 の手順を以下に示す.

- ステップ 1: 大貧民 AI5 体による 10000 ゲームからなる 1 セットの累積得点を取得する.
- ステップ 2: TrueSkill の環境パラメータ設定する. ゲームのランダム性を考慮したパラメータを五ヶ谷らの提案手法よりも小さい β を設定する.
- ステップ 3: ステップ 1 で得られた累積得点に対して、点数の高いものから 1~5 の順位を割り当て、試合の順位リストを作成する.
- ステップ 4: 順位リスト、プレイヤーの現在のスキル、環境パラメータを TrueSkill に入力し、レートを更新する.
- ステップ 5: 行ったセット数分だけ、上記ステップを繰り返す.

5.3 提案手法 3: 引き分けを考慮してレートを算出する手法

五ヶ谷らの提案手法においては、累積得点が等しいプレ

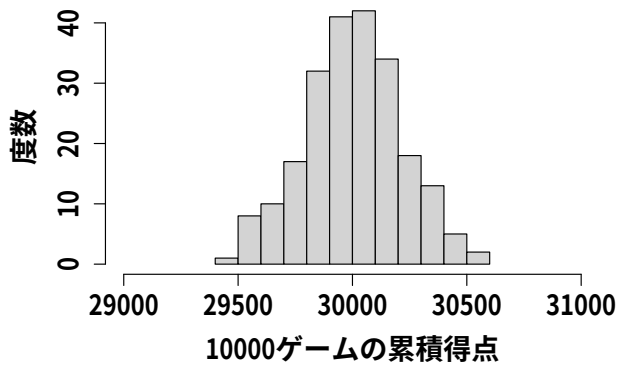


図 1 default5 体を 10000 ゲーム、1000 セット対戦させた default1 体の累積得点のヒストグラム

表 1 default5 体を 10000 ゲーム、1000 セット対戦させた default1 体の累積得点のヒストグラムにおける基本統計量

	最小値	中央値	最大値	平均値	標準偏差
基本統計量	29445	30017	30579	30006	212

イヤを引き分けとカウントし、1 点でも大きければ、そのプレイヤーを勝利と判断していた。しかしながら、コンピュータ大貧民は多人数不完全情報ゲームであり、ランダム性の強いゲームである。つまり、数点の差はプログラムの強さに依存するものではなく、ランダム性によるものの可能性がある。そこで、累積得点が等しいプレイヤーを引き分けとするのではなく、一定の差の範囲内ならば引き分けと解釈することを提案する。TrueSkill において、引き分けは、勝ち負け以上に情報があるとしてプレイヤーのスキルの更新が行われるため、コンピュータ大貧民の試合結果を適切に解釈することは、より適切なレート算出に繋がると考えられる。

この手法を用いるためには、引き分けと解釈する点数を決める必要がある。ここで、大貧民 AI の default5 体を 10000 ゲーム、1000 セット対戦させた default1 体の累積得点のヒストグラムを図 1 に示す。また、図 1 の基本統計量を表 1 に示す。表 1 の標準偏差を見ると、同じ大貧民 AI でも、10000 ゲーム終了時点の得点に 200 点程差が出ることがわかる。同じ大貧民 AI 同士ならば引き分けを起こすと考えたとすると、大貧民 AI の得点差が 200 点程度あれば、その大貧民 AI 同士は引き分けを起こしていると思えることができる。そこで本提案手法では、200 点差は引き分けであると解釈する。提案手法 3 の手順を以下に示す。

ステップ 1: 大貧民 AI5 体による 10000 ゲームからなる 1 セットの累積得点を取得する。

ステップ 2: ステップ 1 で得られた累積得点に対して、点数の高いものから 1~5 の順位を割り当てる。ただし、累積得点の差が 200 点以内のプレイ

ヤはそれぞれ同じ順位を割り当てる。

ステップ 3: TrueSkill の環境パラメータを設定する。

ステップ 4: 大貧民 AI5 体の順位リストと、環境パラメータを TrueSkill に入力し、レートを更新する。

ステップ 5: 行ったセット数分だけ、上記ステップを繰り返す。

5.4 提案手法 4: コンピュータ大貧民の得点情報を考慮してレートを算出する手法

TrueSkill2 は、順位以外の統計情報をレート算出に使用することで、TrueSkill よりもレートの精度を向上させた。同様に、コンピュータ大貧民の統計情報を取り入れることを考える。大貧民 AI においては、弱い AI は非常に弱く、負け続ける傾向があることが経験的に知られている。そこで、大貧民 AI の強弱の基準得点を設定し、1 セットでのプレイヤーの累積得点が基準以下ならば、そのプレイヤーのスキルは低いと見積もる手法を提案する。

本手法を適用するためには、基準の得点を設定する必要がある。そこで、五ヶ谷らの提案手法で用いられた対戦データにおける 38 体の大貧民 AI の得点データの最小値と最大値を調べた。結果を図 2 に示す。このとき図 2 の横軸の大貧民 AI は五ヶ谷らの手法により求めたレートの降順に並んでいる。また、すべての対戦における各大貧民 AI の累積得点を、ヒストグラム図 3 に示す。図 2 を見ると各大貧民 AI 毎に取り得る得点の範囲がある程度決まっていることがわかる。図 2 の左側を見ると強い大貧民 AI の最小値は 30000 点を下回っているものが少ないなどの特徴が見られる。このとき、1 試合の結果から順位の情報以外に、一定の得点より低いプレイヤーはスキルを低く見積もることを考える。試合中、各ゲームで貧民を取り続けると、10000 ゲームで 20000 点となる。そこで、スキルの低いプレイヤーを「貧民を 10000 ゲームとった場合の得点である 20000 点を下回るプレイヤー」として考える。図 2 を見ると、レートの低いプレイヤーにその傾向が見られることわかる。ここで、図 3 を見ると、全プレイヤーのスコアは 34000 点から 37000 点程度である。つまり、今回使用した大貧民 AI の標本は、全体的にスキルの低いプレイヤーが多く、スキルの高いプレイヤーが少ない傾向にある可能性が高い。そのため、20000 点をそのまま基準とすると、弱い大貧民 AI を含み切れない可能性がある。そこで、スキルの低いプレイヤーの基準を、図 3 における最頻値と平均値のズレの分である 5000 点、20000 点に加算する。以上のことから、スキルの低いプレイヤーの基準を 25000 点以下のプレイヤーとする。

次に、TrueSkill にスキルの低いプレイヤーを表す値を加える必要がある。具体的には、TrueSkill のパフォーマンスファクターにスキルの低いプレイヤーを表すパラメータ *weak_skill* を追加する。TrueSkill の内部計算の中では、式 2 の尤度 $P(r|s_i)$ で表される確率分布の 1 つに式 3 がある。

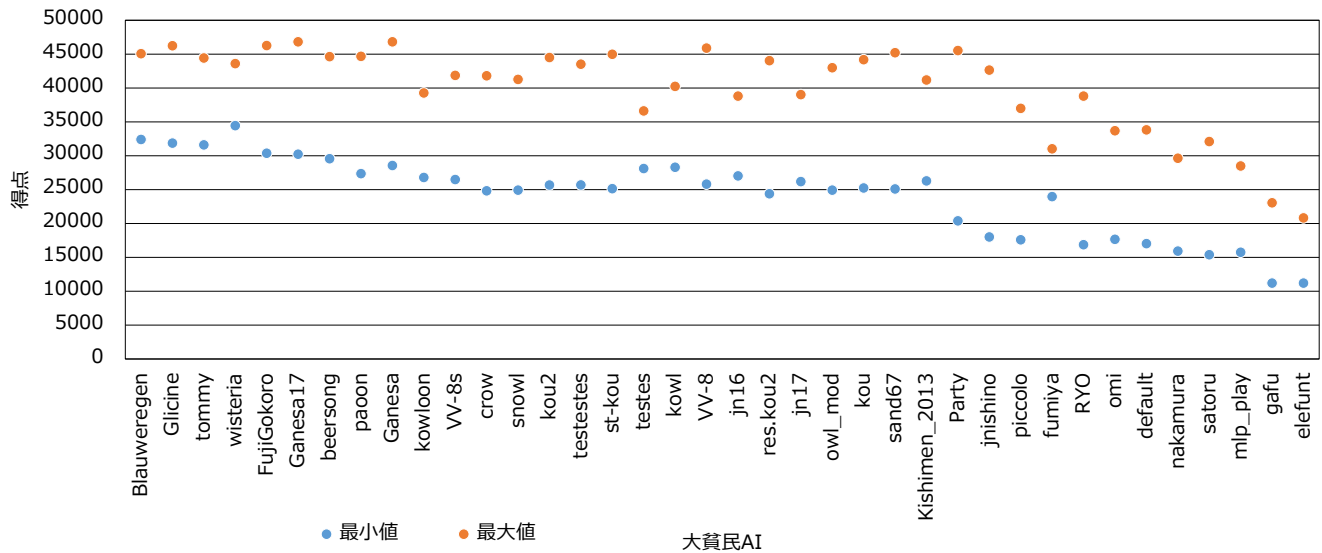


図 2 大貧民 AI の得点の最大値と最小値

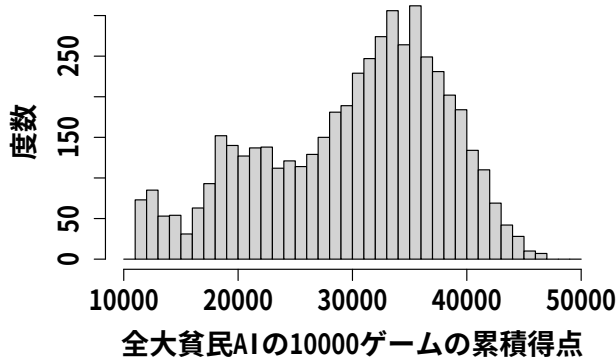


図 3 大貧民 AI18 体のすべての対戦における累積得点のヒストグラム

3 に $weak_skill$ を追加することで、式 4 となる。ここで、 p_i はパフォーマンスを表す確率変数である。

$$P(p_i | s_i) = N(p_i; s_i, \beta^2) \quad (3)$$

$$P(p_i | s_i) = N(p_i; s_i + weak_skill, \beta^2) \quad (4)$$

コンピュータ大貧民の得点情報を考慮した提案手法 4 の手順を以下に示す。

- ステップ 1: 大貧民 AI5 体による 10000 ゲームからなる 1 セットの累積得点を取得する。
- ステップ 2: ステップ 1 で取得した累積得点をもとに、ある基準値より低いプレイヤーに負のレートを決して $weak_skill_list$ を作成する。これは、順位リストと 1 対 1 対応である。
- ステップ 3: ステップ 1 で得られた累積得点に対して、点数の高いものから 1~5 の順位を割り当てる。

ステップ 4: TrueSkill の環境パラメータを設定する。

ステップ 5: 大貧民 AI5 体の順位リストと、環境パラメータ、そして、 $weak_skill_list$ を TrueSkill に入力し、レートを更新する。

ステップ 6: 行ったセット数分だけ、上記ステップを繰り返す。

6. 実験と考察

6.1 実験の概要

計算機実験により、各種提案手法の有効性を評価する。実験では、4 つの提案手法を段階的に追加し、追加前と追加後の精度を比較する。各比較により、各提案手法の有効性を明らかにする。

各実験で使用する大貧民 AI は、五ヶ谷らが実験で用いたものと同じとした。また、精度の評価には、レートと、標準偏差から求められる予測勝率と、実際勝率との平均二乗誤差 (以降、単に平均二乗誤差と記す) を使用する。ここで、各手法を区別しやすいように、以下のように表記する。

TS_Gokaya TrueSkill を用いた五ヶ谷らのレーティング手法

TS_Multi TS_Gokaya に、複数人対戦の結果としてレートを算出することを取り入れた手法。提案手法 1 を取り入れたものに相当する。

TS_Beta TS_Multi に、ゲームのランダム性を調整することを取り入れた手法。提案手法 1 と 2 を取り入れたものに相当する。

TS_Draw 最もレートの精度が高くなった β を使用したときの TS_Beta に、引き分けを考慮することを取り入れた手法。提案手法 1, 2, 3 を取り入れたものに相当する。

TS_Sta TS_Draw に、プレイヤーの統計情報を考慮するこ

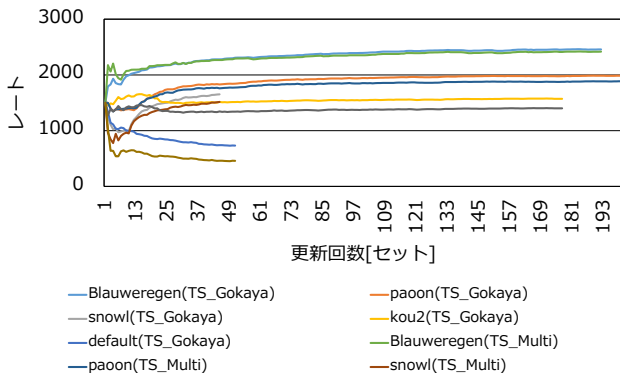


図4 TS_Gokaya と TS_Multi における更新回数とレート

とを取り入れた手法. 提案手法 1, 2, 3, 4 を取り入れたものに相当する.

6.2 実験 1 TS_Multi

実験 1 では, コンピュータ大貧民の対戦結果を複数人対戦として解釈してレートを算出することの有効性を確認する. この実験 1 で比較する対象は, TS_Gokaya と, TS_Multi である. TS_Gokaya と TS_Multi の平均二乗誤差を比較することにより, レートの精度が向上しているか否かを確認する. 実験 1 の結果として, 算出された TS_Multi のレートと標準偏差を表 2 に示す. TS_Gokaya と TS_Multi の平均二乗誤差を表 3 に示す. また抜粋した大貧民 AI における TS_Gokaya と TS_Multi のレートと標準偏差の変遷を図 4, 図 5 に示す.

図 3 を見ると, TS_Multi の平均二乗誤差の平均は, TS_Gokaya に比べ, 小さくなっていることが分かる. つまりレートの予測精度が向上したといえる. また, 図 5 を見ると, TS_Multi の方が, 収束が早いことがわかる. これは, 対戦データとして 5 人の順位リストを渡したことにより, ファクターグラフの比較因子での計算回数が多くなり, 正規分布の近似が適した値になった可能性が考えられる. つまり, 複数人対戦のゲームを 2 人ゲームとして解釈することは, 順位リストを独立した勝ち負けのデータに分割してしまうため, 対戦データから得られる情報を意図的に少なくしていたことが考えられる. このことから順位という情報をそのまま扱うことが, コンピュータ大貧民でレートを算出する上で適しているといえる.

以上のことから, コンピュータ大貧民の対戦結果を複数人対戦として解釈してレートを算出することは, レートの精度向上が見られるため, 提案手法 1 は有効であるといえる.

6.3 実験 2 TS_Beta

実験 2 では, コンピュータ大貧民においてランダム性を考慮するパラメータ β を小さく設定することは, レートの精度の向上に有効か否かを明らかにする. TS_Beta は,

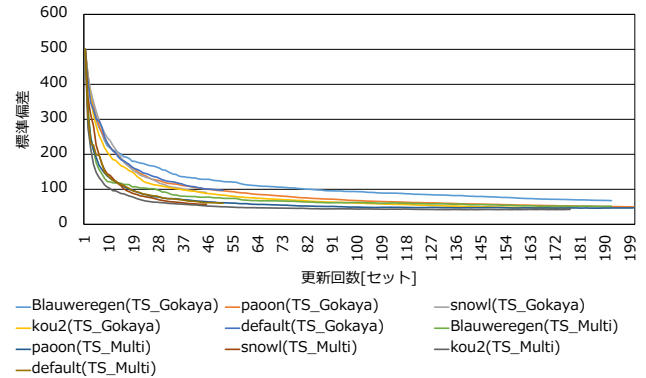


図5 TS_Gokaya と TS_Multi における更新回数と標準偏差

TS_Multi に提案手法 2 を取り込んだものであるため, 実験 2 では, TS_Multi と TS_Beta におけるレート, 標準偏差, 平均二乗誤差を比較する. 設定する β は五ヶ谷らの提案手法よりも小さい $\beta = \{50, 100, 150\}$ とした. 最もレートの精度が向上した $\beta = 50$ のときのレートと標準偏差を表 2 に示す. また, $\beta = \{50, 100, 150, 200\}$ の各 β の時の平均二乗誤差の平均を表 4 に示す. ここで, $\beta = 200$ のときは, TS_Multi と同じレーティング手法になる.

表 4 を見ると, β の値が小さくなるにつれ, 平均二乗誤差が小さくなっていることがわかる. その中でも $\beta = 50$ の平均二乗誤差が小さく, その時の精度が高いことがわかる. これは, コンピュータ大貧民を 10000 ゲーム 1 セットとして対戦させた結果は, 将棋や囲碁のような実力ゲームに近い小さなランダム性を持つことが示せたともいえる.

以上のことから, コンピュータ大貧民においてランダム性を考慮するパラメータ β を小さく設定することは, レートの精度を向上させる. 特に, その中でもより小さい β を設定したとき, レートの精度の向上が見られるため, 提案手法 2 は有効であるといえる.

6.4 実験 3 TS_Draw

実験 3 では, コンピュータ大貧民における累積得点の僅差を引き分けと解釈することは, レートの精度の向上に有効か否かを明らかにする. プレイヤ間の累積得点が 200 点差以内なら引き分けと見なして順位リストを作り, この情報をもとにレート, 標準偏差, 平均二乗誤差を算出する. TS_Draw では, TS_Beta に提案手法 3 を取り込んだものであるため, 実験 3 では, TS_Beta と, TS_Draw のレート, 標準偏差, 平均二乗誤差を比較する. TS_Beta において最も精度が高くなった $\beta = 50$ のときと, TS_Draw の各大貧民 AI の平均二乗誤差とを比較したグラフを図 6 に示す. また, TS_Draw により求められたレートと標準偏差を表 2 に, 平均二乗誤差を表 3 に示す.

表 3 を見ると TS_Beta の $\beta = 50$ のときの平均二乗誤差に比べ, TS_Draw の平均二乗誤差が小さくなっていることがわかる. 図 6 を見ると, st-kou や, kou2, VV-8 などの平

表 2 各実験により算出されたレートと標準偏差 (一部抜粋)

大貧民 AI	TS.Gokaya		TS.Multi		TS.Beta		TS.Draw		TS.Sta	
	レート	標準偏差	レート	標準偏差	レート	標準偏差	レート	標準偏差	レート	標準偏差
Blauwereggen	2458	67.8	2417	37.5	2026	15.2	1889	12.5	1889	12.5
tommy	2351	105.4	2381	60.5	2052	23.8	1885	19.3	1885	19.3
paoon	1986	50.0	1885	29.6	1617	17.9	1585	15.3	1582	15.3
wisteria	2240	194.6	2293	125.8	1917	61.5	1837	48.3	1836	48.4
kou2	1571	45.8	1400	25.9	1141	8.1	1100	7.1	1085	7.1
snowl	1651	91.0	1515	53.4	1263	24.9	1197	21.4	1187	21.9
jn16	1432	87.5	1254	48.5	1090	15.7	1065	13.8	1050	14.0
jnishino	997	45.3	777	25.7	806	11.1	816	10.4	643	17.4
default	732	97.2	458	55.9	627	21.7	655	19.6	299	24.7
elefunt	105	79.4	-249	47.2	237	27.8	254	27.9	-183	28.4

表 3 各実験により求められた平均二乗誤差の比較

大貧民 AI	TS.Gokaya	TS.Multi	TS.Beta	TS.Draw	TS.Sta
Blauwereggen	0.0047	0.0029	0.0004	0.0016	0.0016
Glicine	0.0012	0.0021	0.0036	0.0029	0.0029
Ganesa17	0.0086	0.0023	0.0001	0.0001	0.0000
FujiGokoro	0.0110	0.0032	0.0013	0.0001	0.0001
tommy	0.0061	0.0018	0.0040	0.0024	0.0024
beersong	0.0021	0.0007	0.0000	0.0007	0.0007
paoon	0.0022	0.0007	0.0000	0.0007	0.0007
Ganesa	0.0049	0.0025	0.0000	0.0000	0.0000
wisteria	0.0124	0.0054	0.0013	0.0020	0.0020
VV-8s	0.0189	0.0158	0.0089	0.0080	0.0082
kowloon	0.0361	0.0297	0.0039	0.0028	0.0022
st-kou	0.0063	0.0044	0.0047	0.0019	0.0018
kou2	0.0086	0.0079	0.0096	0.0036	0.0033
testestes	0.0088	0.0073	0.0083	0.0086	0.0098
crow	0.0236	0.0179	0.0050	0.0042	0.0041
snowl	0.0372	0.0277	0.0014	0.0022	0.0018
VV-8	0.0024	0.0017	0.0049	0.0015	0.0018
res.kou2	0.0030	0.0043	0.0142	0.0076	0.0091
testes	0.0167	0.0144	0.0174	0.0185	0.0194
kowl	0.0150	0.0128	0.0126	0.0181	0.0189
jn16	0.0120	0.0131	0.0179	0.0101	0.0103
jn17	0.0134	0.0132	0.0164	0.0166	0.0162
owl_mod	0.0020	0.0012	0.0052	0.0039	0.0048
kou	0.0042	0.0030	0.0019	0.0013	0.0016
Kishimen_2013	0.0115	0.0093	0.0024	0.0038	0.0011
sand67	0.0102	0.0083	0.0020	0.0033	0.0010
Party	0.0099	0.0078	0.0005	0.0006	0.0000
fumiya	0.0130	0.0124	0.0150	0.0136	0.0000
jnishino	0.0061	0.0048	0.0002	0.0003	0.0000
piccolo	0.0141	0.0125	0.0103	0.0094	0.0000
default	0.0051	0.0044	0.0005	0.0003	0.0008
RYO	0.0071	0.0048	0.0002	0.0004	0.0000
omi	0.0046	0.0038	0.0004	0.0001	0.0008
nakamura	0.0051	0.0043	0.0001	0.0001	0.0000
satoru	0.0025	0.0016	0.0000	0.0000	0.0000
mlp_play	0.0021	0.0012	0.0000	0.0000	0.0000
gafu	0.0017	0.0008	0.0000	0.0000	0.0000
elefunt	0.0003	0.0001	0.0000	0.0000	0.0000
平均	0.0092	0.0072	0.0046	0.0040	0.0034

表 4 各 β における平均二乗誤差の平均

β	平均二乗誤差の平均
50	0.0046
100	0.0051
150	0.0061
200	0.0072

均二乗誤差が大幅に小さくなっている。これらの大貧民 AI は、200 点以内の点数差が多かった。つまり、引き分けを考慮することにより、より精度の高いレートの算出ができた

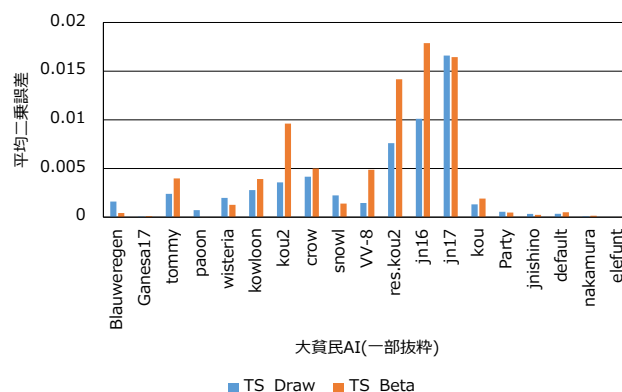


図 6 各大貧民 AI(一部抜粋) における TS.Draw と TS.Beta の平均二乗誤差の比較

と考えられる。多くのレーティングシステムでは考慮されていない引き分けであるが、引き分けを考慮することは勝ち負けよりも重要であると考えられる。

以上のことから、コンピュータ大貧民における累積得点の僅差を引き分けを解釈することは、レートの精度の向上が見られるため、提案手法 3 は有効であるといえる。

6.5 実験 4 TS_Sta

実験 4 では、コンピュータ大貧民の得点から得られる情報を利用することは、レートの精度の向上に有効か否かを明らかにする。スキルの低いと推測されたプレイヤーに与える負のバイアスは $WeakSkill = -50$ とした。TS_Sta は、TS.Draw に提案手法 4 を取り込んだものであるため、実験 4 では、TS.Draw と、TS_Sta におけるレート、標準偏差、平均二乗誤差を比較する。TS_Sta を用いて算出されたレートと標準偏差を表 2 に示す。また、各大貧民 AI(一部抜粋) における TS.Draw と TS_Sta の平均二乗誤差を図 7 に示す。

表 2 より、TS_Sta は、TS.Draw よりも精度が向上していることが分かる。図 7 を見ると、Kishimen_2013, sand67, fumiya, jnishino など精度の向上が見られる。これらはレートで見るとスキルの低いプレイヤーである。試合ごとの得点情報を用いることで、スキルの低いプレイヤーのレート

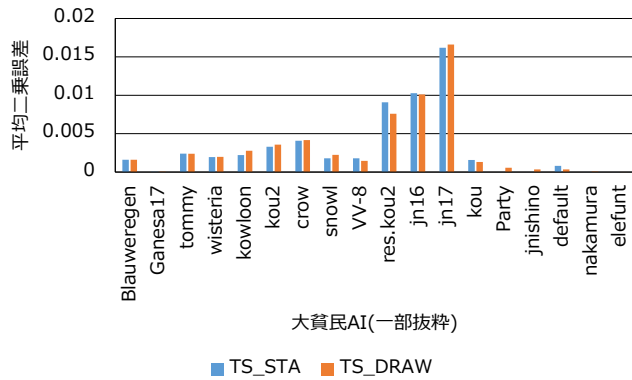


図 7 各大貧民 AI (一部抜粋) における TS_Draw と TS_Sta の平均二乗誤差の比較

の予測精度が向上したことが考えられる。

以上のことから、コンピュータ大貧民の得点から得られる情報を利用することは、レート精度の向上が見られるため、提案手法 4 は有効であるといえる。

6.6 考察

実験 1, 2, 3, 4 では、各提案手法を用いたときの平均二乗誤差を算出した。この中で、特に平均二乗誤差の平均の変化が大きかったのは、 $\beta = 50$ にしたときのレート算出結果である。コンピュータ大貧民の実際勝率を見ると勝率 100% のものと 0% のものがほとんどである。このような実際勝率の傾向を予測率に反映するためには、少しのレートの変化で勝率が変化するようなモデルを考えることが挙げられる。このように、少しのレートの変化で勝率が変化するゲームは一般的に将棋や囲碁のような実力ゲームである。従来の手法の β では、コンピュータ大貧民の 10000 ゲームを 1 セットとするゲーム性が、実力ゲームに近いゲーム性であることを表現できていなかったため、今回、適切な β の設定により、レートの最適化ができたと考えられる。また、実験 3, 4 の結果については、平均二乗誤差の平均は小さくなったものの、変化量が小さかった。これらの原因として、いくつかの点が考えられる。実験 3 については、引き分けの基準値の設定の仕方が考えられる。コンピュータ大貧民の全大貧民 AI における取得点数の振動、もしくは大貧民 AI ごとに引き分けの基準値が動的に動くような設定をすればより精度は向上するかもしれない。実験 4 については、平均二乗誤差の精度が高い大貧民 AI がレートの下位に集中していたため、精度の向上が小さかったのではないかと考えられる。どの実験結果の平均二乗誤差を見てもレートを降順に並べたときの中央が高い傾向にある。そのため、中くらいのスキルを持つプレイヤーの特徴を、ゲームの統計情報から得られれば精度はより向上するのではないかと考えられる。

7. おわりに

本研究では、コンピュータ大貧民特有の性質を考慮した 4 つのレーティング手法を提案した。それぞれ、複数人対戦としてレートを算出する手法、ゲームのランダム性を考慮するパラメータ β を調整する手法、引き分けを考慮する手法、プレイヤーの得点情報を考慮する手法である。有効性を検証するために、計算機実験を行った。実験の結果、これらの提案手法を適用することで、レートの精度が段階的に向上していくことが分かった。また、その中でもすべての手法を適用した結果が、五ヶ谷らの手法よりも、実際勝率と予測勝率の平均二乗誤差が大幅に小さくなり、レートの精度が向上した。また、レートの精度だけでなく、レートの収束速度や、標準偏差の収束速度も五ヶ谷らの手法よりも向上することが分かった。本研究により算出された TrueSkill のレートを用いることにより、大貧民 AI の研究者が自身の作成した大貧民 AI の正確な強さを知ることや、更なる強い戦略の解明に繋がることが期待される。

今後の課題としては、より多くの母集団の大貧民 AI を使用することや、別の対戦データを使用することで提案手法の更なる妥当性を調べるのがあげられる。

参考文献

- [1] Junjie Li, Sotetsu, Koyamada, Qiwei Ye, Guoqing Liu, Chao Wang, Ruihan Yang, Li Zhao, Tao Qin, Tie-Yan Liu, Hsiao-Wuen Hon. *Suphx: Mastering Mahjong with Deep Reinforcement Learning*, arXiv:2003.13590v2 [cs.AI], 2020.
- [2] 電気通信大学, UEC コンピュータ大貧民大会. <http://www.tnlab.inf.uec.ac.jp/daihinmin/2020/> (参照 2020-01-23)
- [3] 五ヶ谷純平, 大久保誠也, 若月光夫, 西野哲朗. コンピュータ大貧民におけるレーティング手法について. 情報処理学会研究報告, Vol. 2020-GI-43, No. 16, pp. 1-8, 2020.
- [4] Ralf Herbrich, Tom Minka, and Thore Graepel. *TrueSkill™: A Bayesian Skill Rating System* Advances in Neural Information Processing Systems 20, 2007.
- [5] 電気通信大学. ドキュメント/UEC 標準ルールについて. http://www.tnlab.inf.uec.ac.jp/daihinmin/2020/document_rules.html. (参照 2020-01-23).
- [6] Arpad E. Elo. *The rating of chessplayers, past and present*, Arco Pub. 1978.
- [7] 森田茂彦, 松崎公紀. 大貧民における初期手札の不均等性を考慮したレーティングアルゴリズムの提案. 情報処理学会研究報告, Vol. 2014-GI-31, No. 14, pp 5, 2014.
- [8] Mark E Glickman. *The Glicko system* Boston University, 1995. <http://glicko.net/glicko/glicko.pdf>.
- [9] Mark E Glickman. *Example of the Glicko-2 system* Boston University, 2013. <http://glicko.net/glicko/glicko2.pdf>.
- [10] 豊田秀樹. 基礎からのベイズ統計学：ハミルトニアンモンテカルロ法による実践的入門. 朝倉書店. 2015.
- [11] Minka, Tom and Cleven, Ryan and Zaykov, Yordan, *TrueSkill 2: An improved Bayesian skill rating system* No. MSR-TR-2018-8, 2018.