Vol.2020-GI-43 No.16
2020/3/13

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

五ヶ谷 純平[1]　大久保 誠也[2]　若月 光夫[1]　西野 哲朗[1]

概要：多人数不確定不完全情報ゲームの一つであるコンピュータ大貧民において，プレイヤの強さを定量的に評価する手法は十分に検討されていない．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手法を確立することである．大貧民には，レートの精度を低下させるいくつかの性質がある．そこで，大貧民特有の性質を考慮したレーティング手法を提案する．提案手法では，それらの性質による影響を排除するため，10,000 ゲームの累計得点を用いてレートを更新する．また，汎用的なレーティングシステムと組み合わせて用いることでレートを更新する．提案手法の有効性と性質を明らかにするため，計算機実験による評価を行った．その結果，提案手法は既存手法より高い精度でレートを求められることや，提案手法と TrueSkill の組み合わせが最も優れていること，プレイヤの間に相性関係があること等が明らかとなった．

キーワード：コンピュータ大貧民, レーティングシステム, Elo, Glicko, Glicko-2, TrueSkill

1. はじめに

大貧民は多人数不確定不完全情報ゲームの一つである．コンピュータ大貧民では，大貧民を 5 つのゲーム AI（以降，大貧民 AI）同士で対戦させる．大貧民 AI の強さを競う場として UEC コンピュータ大貧民大会 [1]（以降，UECda）がある．UECda は，2006 年以降，電気通信大学で毎年開催されている．UECda では，同じ年度に参加した大貧民 AI の強さを競う．そのため，数多く存在する大貧民 AI 同士の強さの優劣は，断片的にしか知られていない．また，大貧民 AI の強さを客観的に測る定量的な指標として広く使われている手法はまだ存在しない．そのため，大貧民 AI の強さの客観的な評価手法が求められている．

コンピュータ大貧民には，第 2 節に示すような性質がある．これらの性質によりレートの精度が低下するため，汎用的なレーティングシステムをそのまま適用することは好ましくない．森田ら [2] は，1 ゲーム単位でレートを更新する大貧民 AI のレーティング手法を提案した．森田らの手法では，「初期手札の不均等性」という性質による影響を打ち消すようにレートの変動量を補正する．しかし，その他の性質による影響は打ち消せていない．そのため，コンピュータ大貧民の性質による影響を打ち消せるレーティング手法が求められる．

そこで本研究では，コンピュータ大貧民の性質による影響を排除できるレーティング手法を確立することを目的とする．本研究では，大貧民特有の性質を考慮したレーティング手法を提案する．提案手法では，それらの性質による影響を排除するため，10,000 ゲームの累計得点を用いてレートを更新する．また，汎用的なレーティングシステムと組み合わせて用いることでレートを更新する．提案手法の有効性と性質を明らかにするため，計算機実験による評価を行う．

[1] 電気通信大学大学院 情報理工学研究科
[2] 静岡県立大学 経営情報学部

2. コンピュータ大貧民

大貧民は，トランプカードを用いた日本発祥のゲームである．複数人でプレイする大貧民は多人数不確定不完全情報ゲームである．大貧民は，以下のようなルールでプレイする．

ゲームの単位　下記の手札配布から手札交換，カードの提出を経て，プレイヤ 1 人を残して他のプレイヤの全員が上がるまでを「1 ゲーム」として扱う．大貧民は，複数のゲームを繰り返して遊ぶ．

勝利条件　席順に応じて，各プレイヤが自身の手札から場にカードを提出していく．手札の枚数が 0 枚になったら「上がり」である．

階級　上がった順番が「順位」となり，順位に応じて次のゲームの「階級」が決まる．何人でプレイするかによって「何位ならばこの階級である」という定義は異なる．5 人でプレイする場合は，上位から大富豪，富豪，平民，貧民，大貧民である．

手札配布　ゲームの冒頭に，各プレイヤに対してカードをランダムに配布する．各自に配布されたカードを「初

期手札」と呼ぶ．

手札交換　初期手札が配布されたあと，各プレイヤは各自の階級に応じて他のプレイヤと手札の一部を交換する．大貧民の 1 ゲーム目では，全プレイヤの階級が平民であり，手札交換は行われない．カードには提出しやすいものと提出しづらいものがある．上位のプレイヤは下位のプレイヤに対して渡すカードを任意に選択できるため，手札から提出しづらいカードを排除できる．

コンピュータ大貧民では，各ゲームの上がり順に応じて得点が与えられ，ある一定数のゲームを終えたときの累計得点を競う．この「ある一定数のゲーム」を「セット」と定義する．コンピュータ大貧民においては，大貧民のルールによって生じる以下の性質が知られており，レーティングの際に考慮すべきである．

コンピュータ大貧民の性質 1「初期手札の不均等性」　初期手札がどのようなカードで構成されているかが，そのゲームでの勝ちやすさに対して影響することが知られている [3].

コンピュータ大貧民の性質 2「手札交換の影響」　手札交換を経ることで，上位の階級のプレイヤの手札はより強いカードによって，下位の階級のプレイヤの手札はより弱いカードによって構成されるようになる．これにより，連勝や連敗が発生しやすくなっていることが，実験により知られている [4].

コンピュータ大貧民の性質 3「序盤の得点差」　全員が平民である 1 ゲーム目では性質 2 の影響を受けず，性質 1 の影響を強く受ける．2 ゲーム目以降は，性質 1 に加えて性質 2 の影響を受けるため，連勝や連敗が発生しやすい．よって，1 ゲーム目の初期手札が強かったかどうかが，累計得点に対して影響を及ぼす．

コンピュータ大貧民の性質 4「席順の影響」　大貧民では，席順が勝率に影響を与えることが，実験により知られている [5]．具体的には，手番が自分の 1 つ前のプレイヤが強い場合，自分が勝ちづらくなるということが分かっている．

コンピュータ大貧民の性質 5「強さの変化」　コンピュータ大貧民では，ゲームごとに強さが変化する大貧民 AI がある．UECda-2011 で優勝した大貧民 AI である crow[6] は，差分学習法によって行動選択の評価値を変化させる．すなわち，ゲームごとにプレイスタイルが変化する，つまり，ゲームごとに強さが変化し得ることを意味する．同様にして，相手のプレイスタイルをモデル化する大貧民 AI もまた，ゲームごとに強さが変化し得ると考えられる．

UECda では，性質 1 が累計得点に影響を与えないようにするため，数千から 10,000 ゲームを 1 セットとして，1 セットの累計得点で順位を決める．UECda では，性質 2・性質 3 が累計得点に影響を与えないようにするため，100 ゲームごとに階級をリセットする．また，性質 4 が累計得点に影響を与えないようにするため，3 ゲームごとに席替えを行う．

3. レーティングシステム

レーティングシステムの分野では，用語の表記ゆれが多い．本稿では，用語の表記を本節に示すように統一する．

プレイヤの強さの指標を算出する行為のことを「レーティング」と呼ぶ．レーティングによって得られた各プレイヤの強さの指標を「レート」と呼ぶ．レーティングを行う具体的なアルゴリズムや，それを実装したもののことを「レーティングシステム」と呼ぶ．多くのレーティングシステムでは，「2 プレイヤ間のレートの差」から「2 プレイヤ間の勝率」を算出できるようにレートを定義している．レーティングシステムによって，レートをスカラで表現する場合や，ベクトルで表現する場合，確率密度関数の母数で表現する場合などがある．試合を行うごとに，試合の結果に応じて対戦したプレイヤのレートを更新するレーティングシステムを「漸進レーティングシステム」と呼ぶ．漸進レーティングシステムでは，試合を重ねることでレートを収束させる．多くの漸進レーティングシステムでは，入力として与える試合結果の順序を入れ替えると，最終時点でのレートが変化してしまう．そのため，プレイヤのレートを更新する順序は，試合を行った順序に合わせることが望ましい．

漸進レーティングシステムとしては，Elo rating system [7]（以降，Elo）や Glicko rating system [8]（以降，Glicko），Glicko-2 rating system [9]（以降，Glicko-2），TrueSkill [10] などが有名であり，様々な分野で実際に用いられている．Elo は，最も広く用いられている 2 人ゲーム用レーティングシステムである．Elo では，スカラ R によってレートを表現する．Glicko は，Elo を改良した 2 人ゲーム用レーティングシステムであり，「プレイヤのレートの不確かさ」を考慮する．Glicko では，平均値 r，レーティング偏差（標準偏差と同義）RD で表される確率密度関数によってレートを表現する．Glicko-2 は，Glicko を改良した 2 人ゲーム用レーティングシステムである．Glicko-2 では，平均値 μ，レーティング偏差（標準偏差と同義）ϕ で表される確率密度関数によってレートを表現する．TrueSkill は，Glicko を改良した，多人数ゲームにも適用可能なレーティングシステムである．TrueSkill では，平均値 μ，標準偏差 σ で表される確率密度関数によってレートを表現する．

Elo を始めとしたレーティングシステムでは，任意の 3 人のプレイヤ A, B, C の中における 2 プレイヤ間の勝率の関係が，式 1 に示す関係を満たしている場合においてのみ，レートの精度が保証される．$W(x, y)$ は，プレイヤ x のプレイヤ y に対する勝率を表す．

$$\frac{W(A,C)}{W(C,A)} = \frac{W(A,B)}{W(B,A)} \cdot \frac{W(B,C)}{W(C,B)} \tag{1}$$

式 1 の関係を満たしている（ただし，ある程度の勝率の誤差を許容する）状態のことを「勝率が推移的に求まる」と呼ぶ．逆に，式 1 の関係を満たしていない状態のことを「勝率が推移的に求まらない」と呼ぶ．式 1 を変形すると式 2 が得られる．

$$W(A,C) = \frac{W(A,B) \cdot W(B,C)}{2 \cdot W(A,B) \cdot W(B,C) - W(A,B) - W(B,C) + 1} \tag{2}$$

勝率が推移的に求まる場合，式 2 を用いることで，プレイヤ A, B 間の勝率とプレイヤ B, C 間の勝率からプレイヤ A, C 間の予測勝率 $\widehat{W}(A,C)$ を計算できる．勝率が推移的に求まらない場合，予測勝率 $\widehat{W}(A,C)$ と実際の勝率 $W(A,C)$ の誤差が大きくなることがある．勝率が推移的に求まるかどうかは，対象とするゲームの特性によって左右される．

ほとんどのレーティングシステムにおけるレートは，プレイヤ x, y 間のレートの差からプレイヤ x, y 間の勝率を予測できるように設計されている．Elo におけるレート差から求まる予測勝率を式 3 に，Glicko におけるレート差から求まる予測勝率を式 6 に，Glicko-2 におけるレート差から求まる予測勝率を式 8 に，TrueSkill におけるレート差から求まる予測勝率を式 10 に示す．ここで，erfc は相補誤差関数を指す．

$$\widehat{W}(x,y) = \frac{1}{10^{\frac{R(y)-R(x)}{400}} + 1} \tag{3}$$

$$q = \frac{\ln(10)}{400} \tag{4}$$

$$g(RD) = \frac{1}{\sqrt{1 + \frac{3q^2RD^2}{\pi^2}}} \tag{5}$$

$$\widehat{W}(x,y) = \frac{1}{10^{-g(RD(y))(r(x)-r(y))} + 1} \tag{6}$$

$$g(\phi) = \frac{1}{\sqrt{1 + \frac{3\phi^2}{\pi^2}}} \tag{7}$$

$$\widehat{W}(x,y) = \frac{1}{1 + \exp(-g(\phi(y))(\mu(x) - \mu(y)))} \tag{8}$$

$$\mathrm{CDF}(\Delta\mu, \sigma) = 0.5\mathrm{erfc}\left(-\frac{\Delta\mu}{\sigma\sqrt{2}}\right) \tag{9}$$

$$\widehat{W}(x,y) = \mathrm{CDF}\left(\frac{\mu(x) - \mu(y)}{\sqrt{2\beta^2 + \sigma^2(x) + \sigma^2(y)}}\right) \tag{10}$$

多人数ゲームにおいて Elo, Glicko, Glicko-2 を用いてレートを計算する場合，多人数ゲームの対戦結果を 2 人ゲームの対戦結果として解釈するための変換が必要である．変換後の 2 人ゲームの対戦結果の数は，プレイヤが n 人（$n \geq 3$）いるとき，1 プレイヤあたり $n-1$ 個となる．このとき，レートの更新方法としては主に 2 通りの方式がある．一つは，$n-1$ 個の対戦結果を用いて逐次的にレートを更新する方式（以降，逐次方式）である．もう一つは，$n-1$ 個の対戦結果に対して個別に新しいレートを計算したあと，$n-1$ 個の新しいレートの平均値を取って更新後のレートとする方式（以降，平均方式）である．対戦結果を与える順番を入れ替えた場合，逐次方式で得られるレートにロバスト性が無いが，平均方式ではロバストである．

コンピュータ大貧民においては，大貧民 AI の強さを定量的に評価する手法が求められている．そこで，レーティングを導入する試みがなされている [2]（後述）．コンピュータ大貧民におけるレートは，第 2 節で示したコンピュータ大貧民の性質の影響を受けて精度が低下すると考えられる．例えば，性質 1 によって，プレイヤの強さに関わらず，初期手札が強ければレートが上がりやすく，弱ければ下がりやすくなる．また，性質 2 によって，連勝している間はレートが上がり続け，連敗している間はレートが下がり続けるため，レートが振動する．加えて，性質 3 によって，1 ゲーム目の初期手札が良かったかどうかによって，レートが「プレイヤ本来のレート」よりも高く/低くなる．他にも，性質 4 によって，プレイヤの強さに関わらず，他のプレイヤの強さがレートに対して影響を及ぼす．性質 5 は，プレイヤの強さがゲームごとに変動することを表す．現実世界（例えば，チェスや囲碁，将棋などのプロリーグ）のレーティングでは，人間であるプレイヤの強さが時とともに変化するため，レートが変動することは差し支えがない．しかしながら，コンピュータ大貧民では「ゲームを重ねるごとにプレイスタイルが変化していくことを含めた，行動選択のアルゴリズムの出来栄え」を定量的に評価したい．

森田ら [2] は，大貧民における性質 1 による影響を補正するレーティング手法を提案し，いくつかの大貧民 AI のレートを調べた．森田らの手法は，前述の Elo をベースとして，2 つの改良を加えたものである．森田らの手法では，毎ゲーム，レートを更新する．一つは，Elo を多人数ゲームに対応させるための拡張である．森田らの手法では，前述の平均方式を採用している．もう一つは，コンピュータ大貧民の性質 1 による影響を打ち消すための補正である．初期盤面（初期手札が配布された時点）をルートノードとしてモンテカルロ法によるプレイアウトを繰り返すことで，各プレイヤの初期盤面勝率を求める．求まった初期盤面勝率を基に，Elo のレート更新式に補正を掛けることで，性質 1 による影響を打ち消している．しかしながら，森田らの手法では性質 2〜性質 5 による影響を打ち消せない．

4. 提案手法

本研究では，2 プレイヤ間の 1 セットにおける累計得点の差に注目したレーティング手法を提案する．10,000 ゲームからなる 1 セット分の対戦結果があるとき，提案手法は以下の手順でレートを更新する．手順 1 における「1 セット分の対戦データ」は，重複のない 5 プレイヤが 10,000 ゲーム対戦した際の情報であり，「誰が対戦したか」と「それぞれの得点は何点か」という情報を含む．各手順の詳細

を示す擬似コードについては後述する．

手順 1: 1 セット分の対戦データから 2 プレイヤ間の「直接対決の結果」に変換する．ここで，任意の 2 つの組み合わせに対し，累計得点の高い方を勝者とする．

手順 2: 手順 2 によって得られた結果を基に，各種のレーティングシステムによってレートを更新する．ここで，プレイヤ 1 人あたり 4 人の相手それぞれに対してレートが求まり，前述の平均方式で得た値を新しいレートとする．また，引き分けの際は，「1 勝した場合のレート」と「2 勝した場合のレート」の平均値を取ることとする．

提案手法には，以下のような利点がある．

- 多人数ゲームの結果を 2 人の間の勝敗と解釈するため，既存の 2 人ゲーム用のレーティングシステムを用いてレートを更新できる
- 多数のゲームの結果を基にレートを更新するため，大数の法則によって性質 1 による影響を排除できる
- 多数のゲームの累計得点を見るため，性質 2 による影響を排除できる
- 100 ゲームごとに階級をリセットするため，性質 3 による影響を排除できる
- 3 ゲームごとに席替えを行うため，性質 4 による影響を排除できる
- 1 セットを終えた時点の累計得点を見るため，性質 5 に当てはまるプレイヤの「10,000 ゲーム対戦したときの強さ」を評価できる
- プレイ履歴を考慮しないため実装しやすい

また，レートが未知のプレイヤ群 P があるとき，P からランダムに 5 プレイヤを選んで対戦させる操作を繰り返すことで，更に以下の利点が得られる．

- 対戦の組み合わせによる影響を排除できる
- 「密な対戦結果」が得られる
- 直接対決の結果が蓄積されるため，勝率が推移的に求まるか否か，三つ巴が存在するか否かを検証できる
- 直接対決の結果を並べることで対戦表（星取表）を生成できる

手順 1 の擬似コードを Procedure 1 に示す．ここで，Procedure 1 内の各識別子の意味は以下の通りとする．

confrontationResults: 1 セット分の直接対決のリスト（リストの要素数は 10 で，要素として「あるプレイヤ i とあるプレイヤ $j(j \neq i)$ の直接対決の結果」を格納される）

getPlayerNames: 当該の対戦データから，対戦に参加した全プレイヤの名称のリストを返す関数

getPlayerTotalScore: 1 つの引数 *name* を取り，当該の対戦データからプレイヤ *name* の累計得点を返す関数

WIN: あるプレイヤ i の累計得点があるプレイヤ $j(j \neq i)$ の累計得点を上回っていることを意味する定数

DRAW: あるプレイヤ i の累計得点とあるプレイヤ $j(j \neq i)$ の累計得点が等しいことを意味する定数

LOSS: あるプレイヤ i の累計得点があるプレイヤ $j(j \neq i)$ の累計得点を下回っていることを意味する定数

Procedure 1 1 セット分の対戦データから直接対決の結果として解釈する手順

Input: 1 セット分の対戦データ *gameData*
Output: 1 セット分の直接対決の結果のリスト *confrontationResults*

```
1: function ITERPRET-CONFRONTATION-RESULTS(gameData)
2:     playerNames ← gameData.getPlayerNames()
3:     confrontationResults ← ∅
4:     for all i ∈ playerNames do
5:         for all j ∈ playerNames \ {i} do
6:             score_i ← gameData.getPlayerTotalScore(i)
7:             score_j ← gameData.getPlayerTotalScore(j)
8:             if score_i > score_j then
9:                 confrontationResults.append(i, j, WIN)
10:            else if score_i = score_j then
11:                confrontationResults.append(i, j, DRAW)
12:            else
13:                confrontationResults.append(i, j, LOSS)
14:            end if
15:        end for
16:    end for
17: end function
```

1 セット分の対戦データを Procedure 1 に与えることで，1 セット分の直接対決の結果が得られる．また，Procedure 1 による操作は並行して行うことができる．

手順 2 の擬似コードを Procedure 2 に示す．ここで，Procedure 2 内の各識別子の意味は以下の通りとする．

*rate*1*vs*1: 3 つの引数 $rate_i, rate_j, winOrDrawOrLoss$ を取り，勝敗 $winOrDrawOrLoss \in$ {WIN, DRAW, LOSS} を元に更新した $rate'_i, rate'_j$ のタプルを返す関数の関数オブジェクト（引き分けは 0.5 勝 0.5 敗として扱い，勝利した場合のレートと敗北した場合のレートの平均値を返す）

rateDictionary: 大貧民 AI の名称をキーとして，該当する大貧民 AI のレートを値に持つ辞書

$name_i$: 当該のプレイヤ i 対 j の直接対決の結果のプレイヤ i の名称

$name_j$: 当該のプレイヤ i 対 j の直接対決の結果のプレイヤ j の名称

EMPTY_DICTIONARY: 空の辞書

containsKey: 1 つの引数 *key* を取り，当該の辞書のキーに *key* が含まれるか否かを返す関数

INITIAL_RATE: 内部レーティングシステムにおけるレートの初期値

winOrDrawOrLoss: 当該のプレイヤ i 対 j の直接対

Procedure 2　1 セット分の直接対決の結果を用いてレートを更新する手順

Input: 内部レーティングシステムの関数オブジェクト *rate*1*vs*1，大貧民 AI の名称をキーとして，該当する大貧民 AI のレートを値に持つ辞書 *rateDictionary*，1 セット分の直接対決の結果のリスト *confrontationResults*
Output: レートが更新された *rateDictionary*

```
function UPDATE-RATING-VALUES(rate1vs1, rateDictionary, confrontationResults)
    name_0 ← confrontationResults[0].name_i
    name_1 ← confrontationResults[0].name_j
    name_2 ← confrontationResults[1].name_j
    name_3 ← confrontationResults[2].name_j
    name_4 ← confrontationResults[3].name_j
    newRatesDictionary ← EMPTY_DICTIONARY
    for i = 0...4 do
        newRatesDictionary[name_i] ← ∅
    end for
    for all result ∈ confrontationResults do
        if not rateDictionary.containsKey(result.name_i) then
            rateDictionary[result.name_i] ← INITIAL_RATE
        end if
        if not rateDictionary.containsKey(result.name_j) then
            rateDictionary[result.name_j] ← INITIAL_RATE
        end if
        oldRate_i ← rateDictionary[result.name_i]
        oldRate_j ← rateDictionary[result.name_j]
        newRate_i, newRate_j ←
            rate1vs1(oldRate_i, oldRate_j, result.winOrDrawOrLoss)
        newRatesDictionary[result.name_i].append(newRate_i)
        newRatesDictionary[result.name_j].append(newRate_j)
    end for
    for i = 0...4 do
        rateDictionary[name_i] ←
            average(newRatesDictionary[name_i])
    end for
end function
```

決の結果（*winOrDrawOrLoss* ∈ {WIN, DRAW, LOSS}）

average: 1 つの引数 *list* を取り，内部レーティングシステムがレートをスカラで表現する場合はリスト *list* 内の各要素の平均値，2 変数 *x*, *y* で表現する場合は (*list* 内の各要素の *x* の平均値, *list* 内の各要素の *y* の平均値) を返す関数

蓄積された各大貧民 AI レートと，Procedure 1 によって得た 1 セット分の直接対決の結果を Procedure 2 に与えることで，当該のセットで対戦した 5 つの大貧民 AI のレートを更新する．なお，多くの漸進レーティングシステムでは，入力として与える試合結果の順序を入れ替えると，最終時点でのレートが変化してしまう．そのため，Procedure 2 による操作を行う場合，対戦が行われた順序と同一の順序で入力として与えなければいけない．本研究では，1 セットの開始時間を基準に順序を決定した．

また，手順 2 では，任意のレーティングシステムを利用できる．本研究では特に，代表的なレーティングシステムである Elo, Glicko, Glicko-2, TrueSkill を用いた手法を提案する．どのレーティングシステムを手順 2 で用いているかを呼び分けるため，本論文では以下のように呼称する．なお，呼称に含まれる S は set-wise（セット単位）を意味する．

- 「SE」: Elo rating system を用いた提案手法
- 「SG」: Glicko rating system を用いた提案手法
- 「SG2」: Glicko-2 rating system を用いた提案手法
- 「STS」: TrueSkill を用いた提案手法

各手法におけるレートの初期値やハイパーパラメータを表 1 に示す．なお，SG では，1 セットを Glicko におけるレーティング期間の単位とする．また，STS では，TrueSkill を 2 人ゲーム用レーティングシステムとして用いる．SG, SG2, STS では，レートは 2 つのパラメータをもつ．そのため，平均方式でレートの平均値を求める際，2 つのパラメータそれぞれにおいて別々に平均値を取ることとする．

表 1　各手法のレートの初期値とハイパーパラメータ

提案手法の名称	手順 2 で用いる手法	レートの初期値	ハイパーパラメータ
SE	Elo	$R = 1500$	$K = 10$ $F = 400$
SG	Glicko	$r = 1500$ $RD = 350$	—
SG2	Glicko-2	$\mu = 1500$ $\phi = 350$	$\sigma = 0.06$ $\tau = 1$
STS	TrueSkill	$\mu = 1500$ $\sigma = 1500/3$	$\beta = 400$ draw probability = 0

5. 計算機実験

提案手法の有効性と，大貧民 AI の特徴を明らかにするために，計算機実験を行った．具体的には，各大貧民 AI のレートはどの程度になるか，SE, SG, SG2, STS の 4 つの提案手法のうち最も精度が高い手法はどれか，コンピュータ大貧民における直接対決の勝率は推移的に求まるか否か，大貧民 AI 同士に三つ巴の関係が存在するか否かを調べた．実験は，以下のステップに沿って行った．なお，ステップ I では，UECda が採用している UEC 標準ルール [11] に則って対戦を行った．

ステップ I:　対戦データを収集

ステップ II:　ステップ I で収集した各対戦データから直接対決の結果に変換

ステップ III:　直接対決の結果を用いてレートを更新

ステップ IV:　直接対決の結果と算出したレートを用いて，提案手法の性能を評価

UECda で公開されている大貧民 AI [12] のうち 38 個を大貧民 AI 群 *P* として，1,008 セット分の対戦データを収

集し，レートの計算を行った．対戦データを収集する際のサーバープログラム（ゲームマスターとして動作する）として tndhms-dev-190726 を用いた．C/C++で書かれている大貧民 AI は，コンパイラのバージョンに依存する部分があるため，コンパイルする際は GCC 9.2.0，8.3.0，7.4.0，6.5.0 の中から，できるだけ新しいものを用いた．Java で書かれている大貧民 AI を動作させる際は JRE 1.8.0_172-internal を用いた．

各大貧民 AI における，レートの更新回数（対戦セット数と等しい）と，直接対決の結果における通算勝率と，各提案手法によって得られたレートを表 2 に示す．比較のために，[2] で示されたレートもあわせて示す．なお，表 2 は提案手法 STS によって得られたレートが大きい順に並べている．

レーティング手法において，どの程度の速度でレートが収束するかは重要な要素である．提案手法 SE によるレート R の変遷を図 1(a) に，提案手法 SG によるレート r の変遷を図 1(b) に，提案手法 SG2 によるレート μ の変遷を図 1(c) に，提案手法 STS によるレート μ の変遷を図 1(d) に示す．

第 3 節で示した通り，レーティング手法を用いて得られたレートから勝率を予測できることが望ましい．そこで，予測勝率と実際の勝率を比較することにより，提案手法の有効性を評価する．提案手法 SE, SG, SG2, STS によってそれぞれ得られた 2 プレイヤのレートから式 3，式 6，式 8，式 10 によって求まる予測勝率と実際の勝率の平均二乗誤差を表 3 に示す．加えて，本研究と [2] の双方でレートが調査された大貧民 AI 群 P_{common} に対して同様に計算した結果も，表 3 に示す．

大貧民 AI の相性や特徴を明らかにするため，対戦表を用いて評価する．対戦表は，テップ II で求めた直接対決の結果における，プレイヤ A のプレイヤ B に対する直接対決の勝数と敗数を並べた表である．提案手法 SE でのレートが大きかった上位 15 個の大貧民 AI P_{strong} における対戦表を表 4 に示す．本研究と森田らの研究の双方でレートが調査された 5 つの大貧民 AI P_{common}（paoon，Party，jnishino，fumiya，default）における対戦表を表 5 に示す．なお，表 4 と表 5 は表 2 と同じ順番で大貧民 AI を並べている．また，表 4 と表 5 では，引き分けの数を省略してある．

表 2　1,008 セット経過後の各大貧民 AI のレート

大貧民 AI の名称	レート更新回数	通算勝率 [%]	各手法により得たレート				
			SE	SG	SG2	STS	森田らの手法
Blauweregen	191	97.1	2222	2289	2334	2458	—
Glicine	182	96.8	2203	2252	2313	2418	—
tommy	52	94.2	1971	2197	2169	2351	—
wisteria	15	95.0	1685	2121	2060	2240	—
FujiGokoro	76	87.5	1959	2091	2105	2206	—
Ganesa17	198	90.0	2011	2053	2078	2184	—
beersong	165	84.1	1927	1922	1964	2029	—
paoon	199	83.7	1928	1884	1961	1986	1803
Ganesa	188	78.6	1848	1793	1854	1876	—
kowloon	24	74.0	1645	1713	1741	1784	—
VV-8s	174	71.0	1735	1651	1715	1704	—
crow	45	64.4	1695	1624	1669	1678	—
snowl	44	61.9	1664	1606	1639	1651	—
kou2	176	61.4	1643	1534	1614	1571	—
st-kou	174	63.6	1660	1529	1615	1564	—
testestes	195	60.3	1657	1526	1615	1564	—
testes	19	56.6	1529	1470	1523	1498	—
VV-8	179	55.7	1623	1435	1548	1459	—
kowl	19	53.9	1510	1433	1484	1451	—
jn16	42	53.0	1534	1409	1478	1432	—
res.kou2	185	58.0	1538	1411	1476	1424	—
jn17	28	50.9	1512	1379	1450	1399	—
owl_mod	184	49.5	1472	1325	1402	1325	—
kou	176	45.2	1441	1267	1359	1254	—
Kishimen_2013	61	41.4	1381	1175	1260	1148	—
sand67	170	39.7	1371	1175	1270	1143	—
Party	172	36.9	1299	1131	1192	1091	1452
jnishino	197	30.7	1233	1052	1117	997	1344
piccolo	190	27.2	1178	986	1051	915	—
fumiya	10	12.5	1386	959	1115	892	1649
RYO	189	23.8	1122	917	980	828	—
omi	160	22.8	1094	864	937	764	—
default	50	16.5	1155	813	912	732	1102
nakamura	181	14.4	1002	757	832	640	—
satoru	194	11.6	898	670	713	528	—
mlp_play	189	7.4	832	577	626	414	—
gafu	159	3.9	757	492	530	300	—
elefunt	188	0.0	679	350	390	105	—
Nakanaka	—	—	—	—	—	—	1665
yupi2	—	—	—	—	—	—	1650
taitai	—	—	—	—	—	—	1463
Party	—	—	—	—	—	—	1363

6. 考察

表 2 より，どの手法でも大貧民 AI のレートの大小関係は類似した結果となった．しかしながら，いくつかの点で特徴的な差異があることも分かった．SE と森田らの手法では，fumiya を Party 以上の強さに評価しているが，提案手法 SG2・STS ではそうではない．また，表 4 より，wisteria は FujiGokoro より下に記述した大貧民 AI に直接対決で負けたことはない．しかしながら，提案手法 SE でのレートは非常に低い．これは fumiya と Party のレートの更新回数（対戦のセット数）が少ないからや，提案手法 SE と森田らの手法は Elo（Glicko，Glicko-2，TrueSkill に比べてレートの収束が遅い）を用いているからではないかと考えられる．

表 2 と表 4 を比較すると，Ganesa より上に記述した範囲では，より上位の大貧民 AI は下位の大貧民 AI に直接対決で負けたことがほとんど無いことが分かる．この点は，

表 3　P と P_{common} におけるレート差から求まる予測勝率と実際の勝率の平均二乗誤差

レーティング手法	P での平均二乗誤差	P_{common} での平均二乗誤差
SE	0.0409	0.0590
SG	0.0222	0.0303
SG2	0.0108	0.0031
STS	0.0092	0.0097
森田らの手法	–	0.0399

各提案手法により得られたレートと合致している．一方，crow と VV-8s を比較した場合，直接対決では crow が勝ち越しており，他の試合でも概ね crow の方が成績が良くなることが予測される．これも，レートの更新回数が少ないためと考えられる．今後，より多くの対戦データを収集することで，レートの順位が逆転する可能性がある．

図 1(a) では，1 セット目から 700 セット目付近までレートの差がほぼ一定の傾きで開いていっているのに対して，図 1(b)，図 1(c)，図 1(d) では 200 セット目付近までに急激にレートの差が開いていることが分かる．また，図 1(a) では，default のレートがなだらかに変動しているのに対して，図 1(b)，図 1(c)，図 1(d) では即座に収束していることが分かる．加えて，図 1(a) では，レートが細かく振動を続けているのに対して，図 1(b)，図 1(c)，図 1(d) ではほとんど振動していないことが分かる．これらのことから，提案手法 SE よりも，提案手法 SG, SG2, STS が優れているレーティングシステムと言える．

表 3 より，STS における平均二乗誤差が最も小さいことが分かる．よって，STS で求めたレートの精度が最も高いと言える．また，SG2, STS は森田らの手法よりも高い精度で勝率を予測できることが分かる．

他にも，プレイヤ A, B 間の勝率とプレイヤ B, C 間の勝率から式 2 によって推移的に求まる予測勝率 $\widehat{W}(A,C)$ と A, C 間の実際の勝率との誤差が，±60 ポイント以上となる状況が存在することが分かった．加えて，Blauweregen, Glicine, tommy の間において，勝率の関係が三つ巴となっていた．具体的には，直接対決の結果において，Blauweregen が Glicine に対して 58%，Glicine が tommy に対して 58%，tommy が Blauweregen に対して 54%勝ち越していた．このことから，1 セットの累計得点の差を見て勝敗を判定する場合，一部のプレイヤ間において勝率が推移的に求まらないこと，一部のプレイヤ間において勝率が三つ巴の関係となることが言える．よって，大貧民 AI の間には相性が存在する可能性が示唆される．

表 5 より，5 つの大貧民 AI すべての 2 プレイヤ間で，勝数と敗数の比が 1:0 か 0:1 となっていることが分かる．表 4 においては，実力が離れているプレイヤ A, B において，A, B 間の勝数と敗数の比が 1:0 か 0:1 となっていることが分かる．つまり，セット単位の累計得点の差によって直接対決の結果を判定する場合，強さが大きく異なる 2 プレイヤ間において強い方が弱い方に対して必ず勝つと言える．

表 5　P_{common} における対戦表（A の B に対する直接対決の勝敗数）

A の名称	B の名称				
	paoon	Party	jnishino	fumiya	default
paoon	—	58-0	58-0	2-0	14-0
Party	0-58	—	50-0	—	20-0
jnishino	0-58	0-50	—	—	10-0
fumiya	0-2	—	—	—	—
default	0-14	0-20	0-10	—	—

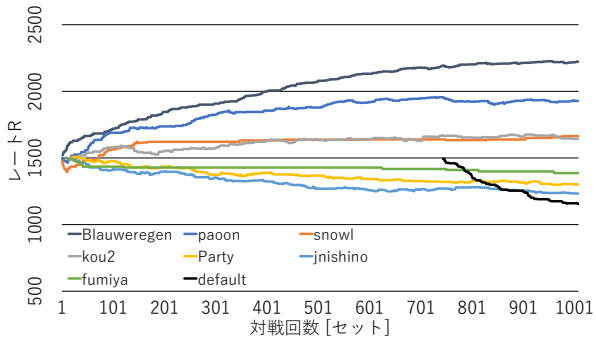
7. おわりに

本研究では，2 プレイヤ間の累計得点差に注目したレーティング手法を提案した．提案手法は，複数のゲームからなる 1 セットを単位とする点や，5 人の結果を 2 人ゲームの結果として解釈する点に特徴がある．また，既存の 2 人ゲーム用のレーティングシステムを用いてレートを更新できる．38 個の大貧民 AI を対象にして計算機実験を行った結果，内部に Glicko-2 を用いた提案手法 SG2 と，内部に TrueSkill を用いた提案手法 STS は，森田らの手法よりも高い精度でレートを算出できることが分かった．ならびに，提案手法 STS によって求まるレートの精度が最も高いことが分かった．他にも，一部の大貧民 AI の間では勝率が推移的に求まらないこと，Blauweregen, Glicine, tommy の強弱の関係は三つ巴の状態になっていること，強さが大きく異なる 2 プレイヤ間においては強い方が弱い方に対して必ず勝つことが分かった．

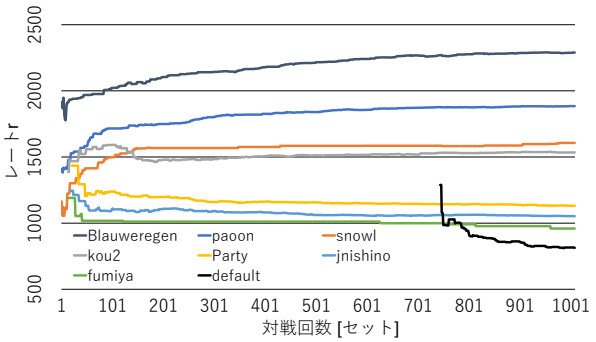
今後，本研究により求まった各レートを用いて，大貧民 AI の特性や戦略と強さとの関係に関する研究に繋がることが期待される．また，大貧民 AI を開発する際，提案手法を用いることで開発した大貧民 AI の性能を評価できる．プレイヤ間の強弱関係において三つ巴が存在するゲームにおいては，プレイヤの「多元的な強さ」を表現できる multidimensional Elo rating [13] や，大渡らの手法 [14] などのレーティングシステムを用いてレーティングを行うことが望ましい．今後，これらのレーティングシステムを用いることで，レートの精度をより高めることが期待される．他にも，TrueSkill 2 [15] などの，より先進的なレーティングシステムを用いた場合の効果についても検討したい．本研究では，10,000 ゲームを 1 セットとして定義したが，10,000 ゲームの対戦には長い計算時間を要する．そのため，より少ないゲーム数で同等の精度が出せるかどうかについても検討したい．

参考文献

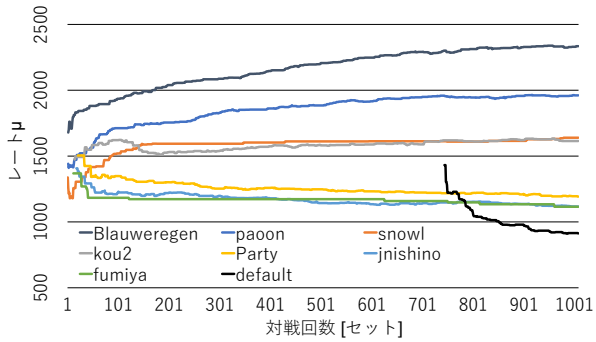
[1] 電気通信大学：UEC コンピューター大貧民大会，http://www.tnlab.inf.uec.ac.jp/daihinmin/. (参照 2020-01-27).
[2] 森田茂彦，松崎公紀：大貧民における初期手札の不均等性を考慮したレーティングアルゴリズムの提案，情報処



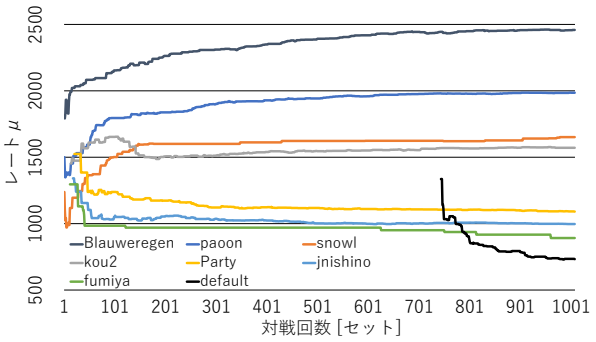
(a) SE における R の変遷



(b) SG における r の変遷



(c) SG2 における μ の変遷



(d) STS における μ の変遷

図 1　各提案手法におけるレートの変遷（抜粋）

表 4　P_{strong} における対戦表（A の B に対する直接対決の勝敗数）

A の名称	B の名称														
	Blauweregen	Glicine	tommy	wisteria	FujiGokoro	Ganesa17	beersong	paoon	Ganesa	kowloon	VV-8s	crow	snowl	kou2	st-kou
Blauweregen	—	36-26	10-18	2-0	32-0	72-0	50-0	62-0	68-0	8-0	46-0	22-0	20-0	56-0	52-0
Glicine	26-36	—	14-10	4-0	28-0	50-0	48-0	62-0	42-0	6-0	56-0	24-0	28-0	56-0	52-0
tommy	18-10	10-14	—	—	12-0	12-0	8-0	28-0	4-0	4-0	4-0	12-0	16-0	14-0	10-0
wisteria	0-2	0-4	—	—	2-0	4-0	2-0	8-0	4-0	4-0	—	—	2-0	4-0	—
FujiGokoro	0-32	0-28	0-12	0-2	—	20-2	12-0	24-0	22-0	2-0	20-0	2-0	2-0	20-0	20-0
Ganesa17	0-72	0-50	0-12	0-4	2-20	—	68-0	44-0	62-0	6-0	60-0	8-0	8-0	76-0	48-0
beersong	0-50	0-48	0-8	0-2	0-12	0-68	—	32-22	58-0	4-0	42-0	16-0	10-0	50-0	50-0
paoon	0-62	0-62	0-28	0-8	0-24	0-44	22-32	—	62-0	4-0	68-0	24-0	28-0	58-0	64-0
Ganesa	0-68	0-42	0-4	0-4	0-22	0-62	0-58	0-62	—	4-0	60-0	4-0	12-0	60-0	52-0
kowloon	0-8	0-6	0-4	0-4	0-2	0-6	0-4	0-4	0-4	—	—	0-4	0-4	4-0	6-0
VV-8s	0-46	0-56	0-4	—	0-20	0-60	0-42	0-68	0-60	—	—	2-4	0-2	40-14	50-6
crow	0-22	0-24	0-12	—	0-2	0-8	0-16	0-24	0-4	4-0	4-2	—	8-12	16-0	12-0
snowl	0-20	0-28	0-16	0-2	0-2	0-8	0-10	0-28	0-12	4-0	2-0	12-8	—	4-0	10-0
kou2	0-56	0-56	0-14	0-4	0-20	0-76	0-50	0-58	0-60	0-4	14-40	0-16	0-4	—	20-26
st-kou	0-52	0-52	0-10	—	0-20	0-48	0-50	0-64	0-52	0-6	6-50	0-12	0-10	26-20	—

理学会研究報告，Vol. 2014-GI-31, No. 14, pp. 1–5 (2014).
[3] 門裕太：大貧民を特徴付ける指標と初期手札に関する研究，電気通信大学平成 28 年度 卒業論文 (2017).
[4] 土橋康希：コンピュータ大貧民における階級制度の効果に関する研究，電気通信大学平成 28 年度 卒業論文 (2017).
[5] 大久保誠也：UEC コンピュータ大貧民における席換えルール導入の効果について，経営と情報，Vol. 31, No. 1, pp. 1–7 (2018).
[6] 小沼啓，本多武尊，保木邦仁，西野哲朗：コンピュータ大貧民に対する差分学習法の応用，情報処理学会研究報告，Vol. 2012-GI-27, No. 1, pp. 1–4 (2012).
[7] Elo, A. E.: *The rating of chessplayers, past and present*, Arco Pub. (1978).
[8] Glickman, M. E.: The Glicko system, http://glicko.net/glicko/glicko.pdf (1995). (参照 2020-01-27).
[9] Glickman, M. E.: Example of the Glicko-2 system, http://glicko.net/glicko/glicko2.pdf (2013). (参照 2020-01-27).
[10] Herbrich, R., Minka, T. and Graepel, T.: TrueSkill™: A Bayesian Skill Rating System, *Advances in Neural Information Processing Systems 20*, MIT Press, pp. 569–576 (2007).
[11] 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/2019/document_rules.html. (参照 2020-01-27).
[12] 電気通信大学：UECda/公開資料，http://www.tnlab.inf.uec.ac.jp/daihinmin/2019/participants.html. (参照 2020-01-27).
[13] Balduzzi, D., Tuyls, K., Perolat, J. and Graepel, T.: Re-evaluating evaluation, *Advances in Neural Information Processing Systems*, pp. 3268–3279 (2018).
[14] 大渡勝己，西野順二：相性関係を考慮したレーティングシステム，情報処理学会研究報告，Vol. 2019-GI-41, No. 24, pp. 1–8 (2019).
[15] Minka, T., Cleven, R. and Zaykov, Y.: TrueSkill 2: An improved Bayesian skill rating system, No. MSR-TR-2018-8 (2018).