RBC エージェント Zubat の動的最適化による性能改良
Improving RBC Agent Zubat by Dynamic Optimization

益山　歩音
Ayuto Masuyama
法政大学情報科学部コンピュータ科学科
E-mail: ayuto.masuyama.4p@cis.k.hosei.ac.jp

Abstract

Reconnaissance Blind Chess is an imperfect information game created for testing AI performance. In this game, players cannot see their opponent's pieces directly, which makes the game challenging for AI agents. Previous research developed an agent called Zubat by exploiting the opponent's uncertainty by using moves that confuse the opponent and take calculated risks. However, Zubat uses fixed computational time allocation and static evaluation weights, which limits its effectiveness across different game phases. This paper improves Zubat by dynamic optimization. Its objective is to exploit the opponent's uncertainty by more detailed analysis and effective utilization of Zubat's capabilities. First, we introduce dynamic time allocation with a two-layer approach. The first layer adjusts time for each turn based on the game phase (early, middle, or late) and board uncertainty. The second layer adjusts time to compute between the Analytical Module (AM) for board evaluation and the Risk-Taker Module (RM) for exploring risky moves. Second, we introduce dynamic weight adjustment for the Mediator Module (MM). The weight parameter for RM's evaluation value is adjusted across the three game phases to maximize the effect of risk-taking strategy in the late phase. We evaluate these methods through systematic experiments against strong opponents.

1. はじめに

ボードゲームは複雑なルールがあり一定の目的があるため，現実の問題に応用しやすい．そのため，人工知能分野で価値のある研究とされている．チェスや囲碁などの伝統的なボードゲームでコンピュータは既に人間を凌駕しているが，特定の不完全情報ゲームではいまだ課題が残っている．

Reconnaissance Blind Chess (RBC)もそのような不完全情報ゲームの 1 種である．RBC は AI エージェントの能力をテストする目的で考案され，能力を競うコンテストが開催されている [1]．これまでに，RBC 不確実性に対処するいくつかの手法が提案されている [2] [3]．Czupyt ら [4]は，相手の不確実性を積極的に利用する手法 Zubat を提案した．その考察で，その手法の有効度がゲームの進行度に応じて変化することを示した．

本研究では，RBC を題材とし，相手の予測を混乱させる手を予測，評価する機能を導入したボットを構築する．本研究の目的は，Zubat の有効度をより詳細に解析し利用することで，相手の不確実性を利用する手法を確たるものとすることである．先行研究で示された戦術の有効性の推移をもとに，より効果的にはたらくよう計算時間や評価値の重みを動的に配分する．そのために，これらを変更する機能を持ったエージェント，対戦を繰り返し複数のパラメータ設定を自動評価するスクリプトを作成する．それらを用いて勝率，平均ターン数，計算時間などの統計を収集，比較する．

2. 関連研究

Czupyt ら [4]は，RBC の基本的な戦術である盤面を予測することに加え，自分の手を予測させにくくする，相手の予測が不十分な場合に有効な手を評価する手法を取り入れたエージェント Zubat を作成した．しかし，その計算時間配分の実装は簡易的なものであり，実践を通して得られた知見が生かされていない．本研究では，適切な動的時間配分を行う改良を加える．

現在，RBC で最も戦績が良いエージェントは，通常のチェス用のエンジンである Stockfish を RBC 用にアレンジした StrangeFish2 である．これは，Stockfish の盤面評価関数をもとに，本来のチェスでは指せない手(チェック放置など)に合わせた調整を加えた盤面評価を行う．StrangeFish2 はオープンソースであり，先行研究および本研究でエージェントの作成に利用している．

過去の研究 [1] [4]から，性能の良いエージェントを作るためには，ゲーム中に得られた情報から可能な盤面状態を追跡することが重要とわかっている．また，勝率と可能な盤面状態の数には相関がある．つまり，トップエージェントは不確実性を小さくするような手を選び，その精度が勝率に影響するといえる．

3. 準備

3.1. RBC

RBC は，不完全情報ゲームにアレンジされたチェスである．相手の駒の位置を直接確認できず，各ターンに 3×3 の範囲を確認(センシング)することで盤面の情報を得るという特徴がある．その他にも，駒が取られたことは通知されるが取った駒・取られた駒の種類はわからない，チェックやチェックメイトの概念がなくキングを捕獲することでゲームが終了するといったルールが存在する．1

Supervisor: Prof. Hiroshi Hosobe

ターンにセンシングと駒の移動を順に 1 度ずつ行い，ゲームが進行していく．盤面の状態を正確に把握することが難しく，不確実性が生まれる．先行研究および本研究における不確実性とは，過去のセンシング結果と矛盾しない盤面状態の数の大きさを指す．

3.2. Zubat

Zubat は，センシングモジュール(SM)，分析モジュール(AM)，不確実性最大化モジュール(UM)，リスクテイカーモジュール(RM)，仲介モジュール(MM)の 5 つのモジュールで構成される．SM は，最適なセンシング位置を決定するためのモジュールである．

AM は，強力なチェスエンジンである Stockfish を用いた盤面評価を行うモジュールである．AM は StrangeFish のフレームワークを利用して実装されている．候補手について，それが指せる可能な盤面状態すべてで評価を行い，最良時パターン，最悪時パターン，平均時パターンについて評価値を算出する．

UM は，相手視点での不確実性を増大させる手を探すモジュールである．相手が持っている情報は確認できないため，リカレントニューラルネットワークによって相手が追跡している盤面の数を推論し，増大させる手を評価する．

RM は，相手の応手が正確でない場合，有効な手を探すモジュールである．相手の応手がランダムであると仮定し，Stockfish の盤面評価が大きくプラスになる手を探す．そのために浅いモンテカルロ木探索を用いる．

MM は，AM，UM，RM の 3 つのモジュールで算出された評価値を重みづけして合算し，最終的な着手を決めるモジュールである．このモジュールで使用される評価値の重みβ，γの値は実験を通して 100，0.125 に決定された．

4. 提案手法

4.1. 概要

本研究では，Zubat の計算時間配分と評価値の重みづけを動的に最適化する 2 つの手法を提案する．それぞれのモジュールの有効性は，試合を通して変化する．これは先行研究によって示され，単純には試合の中盤で有効度が高いと分析された．しかし，先行研究の実装では，モジュールの計算時間や評価値の重みは固定値であり，ゲーム状況への適応が不十分であった．本研究の改良によって，試合状況に応じて計算時間の配分や評価値の係数を動的に最適化し，より適切な手の選択を行えるようになると考える．

具体的には，以下の 2 点で改良する．最初に，計算時間の動的配分を提案する．まず，ターン全体の時間配分比率を序盤・中盤・終盤のフェーズに応じて調整する．盤面が複雑化しやすい中盤に計算時間を多く割くことで，性能向上が見込めると考えた．また，AM と RM の時間配分について，自らの追跡している盤面状態が多ければ AM の盤面解析を優先し，相手の追跡している盤面状態が多いと考えられる(UM の評価値が高い)状態では RM の計算を優先する調整を加える．

次に，MM の重み動的調整を提案する．先行研究では、グリッドサーチにより適切な固定値を決定していたが，ゲームフェーズごとに最適化の余地が存在した．先行研究の結果から，RM が高く評価した手は終盤で有効にはたらきやすいことがわかっている．現在，UM の重みにβが使用され，RM の重みに UM の評価値とγの積が使用されている．本手法では，重みβは維持し，重みγをゲームフェーズに応じて 3 段階に調整する．これにより，終盤でのリスクテイカー手法の効果を最大化することを目指す．

4.2. 手順

提案する手法を Zubat エージェントに実装する手順を述べる．まず，動的時間配分は 2 層のアプローチで実現する．第 1 層はターン全体の時間配分，第 2 層は AM (盤面解析)と RM (リスクテイカー)の時間配分である．第 1 層では，ゲームフェーズと盤面の不確実性に応じて，各ターンの計算時間を調整する．現在のターン数tと追跡している盤面数n_bから，時間配分係数を算出する．現在のターン数tを基準に序盤(1〜20 手)，中盤(21〜59 手)，終盤(60 手以降)の 3 つのフェーズに分類し，各フェーズに対応する時間配分係数を設定する．以下はその一例である．

$$m_t = \begin{cases} 1.0 & (t \leq 20) \\ 1.2 & (21 \leq t \leq 59) \\ 0.8 & (t \geq 60) \end{cases}$$

これは，先行研究で Zubat の性能評価に用いられていた区分である．中盤は重要な局面が多いため，標準より多くの時間を配分し，終盤は残り時間を温存するため時間配分を抑える．これを各ターンの始めに計算時間の割り当てに使用する．この値を調整しながら，勝率の高い値を探る．

第 2 層として，AM と RM への時間配分を動的に決定する．UM の計算は RNN の推論であり，MM は線形結合であるため，モジュールが行う計算はほぼ固定時間である．SM はセンシング処理の評価計算を相手のターン中に可能な限り多く行うため，自分のターンでどの程度持ち時間を消費するか調整が難しい．そのため，モジュール間の時間配分の最適化で，AM と RM に注目した．先行研究では，盤面が複雑なほど RM の有効性が上がると考察されていた．しかし，盤面が複雑になりやすい中盤では，エージェントの追跡している盤面数が大きくなるため，AM の計算時間も長くかかるというジレンマがあった．そのため，第 1 層のような単純なフェーズでの配分ではなく，他の変数が必要であると考えた．具体的には，自分の追跡している盤面数が多い場面では，AM の計算時間を長く取り，UM の評価値が高い(相手が多くの盤面を追跡している可能性が高い)場面では，RM の計算時間に補正を加える．これらの動的配分によって，相手の不確実性を予測する機能をより活かせると考えた．予備実験で，同じ手を評価するのにかかる計算時間が AM と RM でおよそ7: 3であったことから，それを基準に比率を固定した設定を基準とし，盤面数によって AM の計算時間が延びる設定，UM の評価値によって最大 10%計算時間が上下する設定の 3 つを用意した．これらの条件で対戦を行い戦績の比較を行う．

次に，MM の動的重み調整は，最終的な着手を決定する際の評価値統合処理を拡張することで実現する．MM は，AM，UM，RM の各モジュールから得られた評価値を重みβ，γで統合し，最終スコアを算出する．

$$y = y_A + \beta \cdot y_U + \gamma \cdot y_U \cdot y_R$$

ここで，y_Aは AM の評価値(センチポーン)，y_Uは UM の評価値(0〜1 の範囲)，y_Rは RM の評価値(センチポーン)である．

本手法では，βを固定したまま，γの値をゲームの進行に応じて動的に変更する関数を導入する．具体的には，現在のターン数を入力として受け取り，序盤では保守的な値，中盤では標準値，終盤では積極的な値を返す段階関数を実装する．以下はその一例である．

$$\gamma_t = \begin{cases} 0.1 & (t \leq 20) \\ 0.125 & (21 \leq t \leq 59) \\ 0.15 & (t \geq 60) \end{cases}$$

先行研究でγは 0.125 が採用されていたため，その近辺で適切な動的配分の調整を探る．より高度なアプローチとして，UM の評価値を考慮した連続的な調整も検討する．この場合，相手の不確実性が高いと推定される局面ではγを増加させ，リスクテイカー手法の影響を強める．βについては，UM が全フェーズで一貫して有効であることから固定値 100 を維持する．

5. 実装

実装には Python を用い，ライブラリとして主に reconchess，PyTorch を使用する．reconchess は，RBC 用に開発された公式のライブラリである．基本的な対戦機能や，棋譜データからリプレイを再生する機能，ボットを公開マッチングサーバに接続する機能などを提供している．PyTorch は，機械学習に使用するためのライブラリである．UM で利用する RNN などのモデルはこの PyTorch を使用して作成する．

Zubat の戦略は公開されているが，ソースコードは非公開であるため，まずは Zubat を再現する必要がある．Zubat の作成は，先行研究と同様にオープンソースである Strangefish を原型とする．Strangefish には，Zubat の SM と AM に当たるセンシング戦略と盤面分析戦略が既に実装されている．このソースコードに追加する形で，相手視点の不確実性を推測する UM，モンテカルロ木探索を行う RM，それぞれの評価値を統合する MM を作成する．reconchess ライブラリで用意されているログ機能とは別に，それぞれのモジュールの評価値を保存する機能を作成し，実験後の結果解析に使用する．また，全体の時間配分を管理しているクラスに変更を加え，それぞれのターンやモジュールに対する時間配分を変更可能にした．これらの変更を加えた後，公開マッチングサーバに接続し，パラメータを変更しながら自動的に対戦を繰り返すスクリプトを作成した．

6. 実験

6.1. 手順

再現した Zubat に本手法の実装を加え，パラメータを変更しながら対戦を繰り返す．パラメータごとの試合結果・評価ログなどを収集し，解析を行う．実験は，公開マッチングサーバに接続し，同様に接続されているランダムなボットとの対戦を行う．各設定それぞれで 30 回の対戦を実施し，勝率，平均ターン数，計算時間などの統計を収集する．

実験は 3 つの段階で実施する．第 1 段階では，時間配分最適化の効果を測定する．均等配分を基準にして，ターン全体の時間配分，モジュール間の時間配分，およびそれらの組み合わせについて，複数のパラメータ設定を評価する．第 1 層であるターン全体の時間配分係数m_tには，表 1 の値を設定した．また，第 2 層として，AM と RM の時間比率を表 2 の通りに設定した．ターン全体の持ち時間から他モジュールの計算時間を引いた時間のうちの AM の計算時間の割合であり，残った時間が RM に割り当てられる．追跡している盤面の数をBとしたとき，その数によって割合が変動する．また，RM 重視の設定では，UM の評価値が 0.8 以上ならば RM の計算時間を+10%，0.5 以下の場合は−10%した．

表 1 時間配分係数の値

	$t \leq 20$	$21 \leq t \leq 59$	$60 \leq t$
変更なし	1.0	1.0	1.0
保守的	1.0	1.1	0.8
中間	1.0	1.2	0.8
積極的	0.8	1.4	0.7

表 2 AM と RM の計算時間比率

	$B \geq 100$	$10 \leq B \leq 99$	$B \leq 9$
AM 重視	0.8	0.7	0.6
配分固定	0.7	0.7	0.7
RM 重視	0.7	0.7	0.7

第 2 段階では，MM 重み動的調整の効果を測定する．固定重みを基準として，ゲームフェーズに応じた段階的調整，不確実性に応じた連続的調整など，複数のアプローチを比較評価する．特に終盤での勝率工法とリスクテイカー戦略の成功率を検証する．表 3 の通りにパラメータγを設定した．

表 3 MM 重みγの値

	$t \leq 20$	$21 \leq t \leq 59$	$60 \leq t$
変更なし	0.125	0.125	0.125
保守的	0.1	0.125	0.15
中間	0.75	0.125	0.175
積極的	0.5	0.125	0.2

6.2. 結果

各パラメータ設定での勝率を図 1 に示す．第 2 層での AM 優先戦略をとった場合が最も勝率が高く，第 1 層で積極的戦略をとった場合が最も勝率が低くなる結果になった．また，第 2 層での時間配分を行ったものが，そうでないものよりも勝率が高かった．消費している計算時

間の推移を図 2 に，負けた試合に限定した場合の推移を図 3 に示す．行われた試合のリプレイデータから真の盤面を作成し，ターンごとの評価値の平均の推移を図 4 に示す．また，RM の配分を変更した第 2 層の時間変更についての RM への影響を図 5 に示す．AM の評価値が，AM の最大値よりも 50 以上下だったものをリスキームーブと定義し，その前後での評価値の差を調べた．これによると，今回の設定でリスキームーブは多くの場合，評価値にあまり影響を与えていないことがわかる．

MM の重みを動的に変更した実験の結果を図 6 に示す．すべての設定で，初期の値よりも勝率が高かった．動的重み変更時のリスキームーブの割合と影響を図 7 に示す．

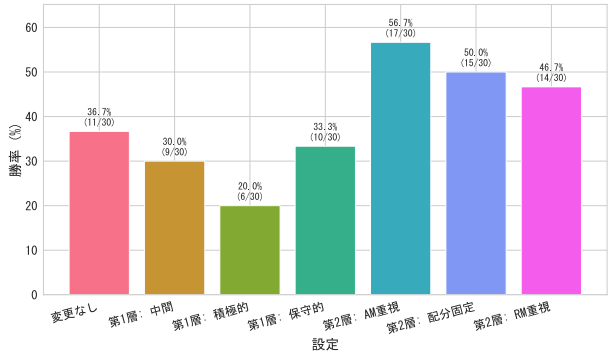

図 1 動的時間配分のパラメータごとの勝率

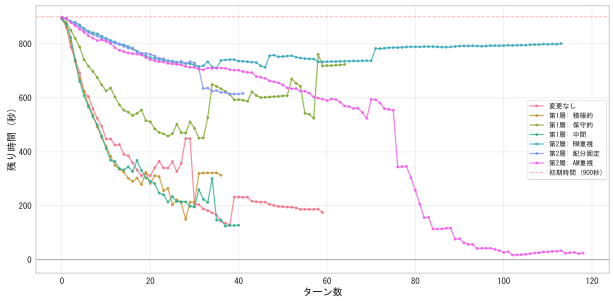

図 2 持ち時間の推移

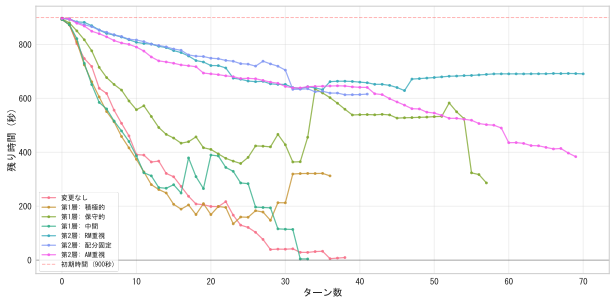

図 3 負け試合における持ち時間の推移

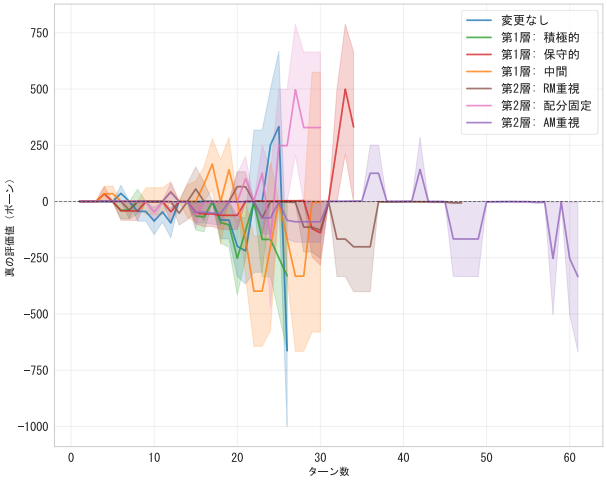

図 4 真の盤面の評価値の推移

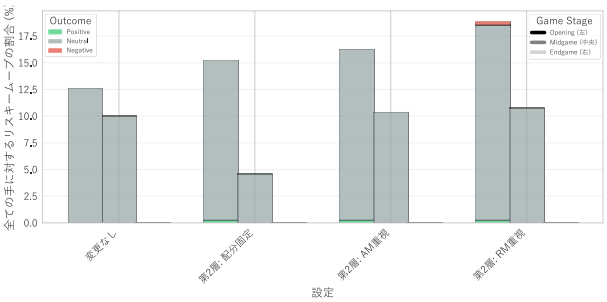

図 5 第 2 層配分変更時のリスキームーブの割合と影響

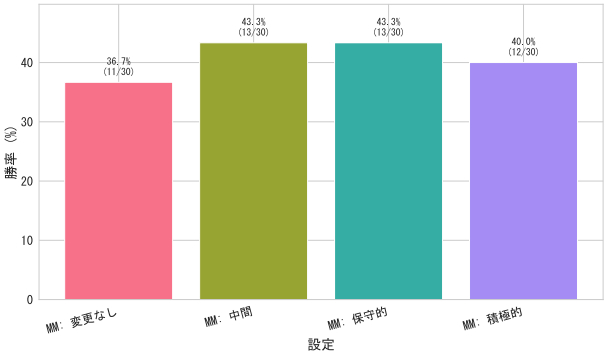

図 6 動的重み変更のパラメータごとの勝率

図 7 動的重み変更時のリスキームーブの割合と影響

7. 議論

現状の実装の勝率は先行研究の Zubat に追い付いていない．これは実行環境の性能差と実装の最適化不足が原因と考えられる．特に AM はエンジン評価を高速に繰り返すため，想定以上に過剰に持ち時間を消費している．図 4 の第一層の設定に注目すると，30 ターンほどで，評価値が大きく下がっていることがわかる．これは，先行研究の実装のままでは，十分な計算能力が無い場合に，序盤で追跡している盤面数が多くなった試合での持ち時間の消費が大きくなりすぎることを示していると考える．そのため，中盤以降に十分な手の検討が行えず，大きく勝率が下がっている．

ターン全体での時間配分を行おうとした第 1 層のパラメータは，すべて上手く機能していない．序盤の計算時間について変更を加えず，最も計算時間がかかる中盤での持ち時間消費を加速させてしまったため，前述の時間不足が起きやすくなってしまったと考える．

第 2 層での時間配分戦略が過剰な時間消費の対策となったことが，勝率が高くなった原因として考えられる．第 2 層の計算時間配分によって，AM の計算時間が最大でターンの7割程度で押さえている．これにより，AMの計算時間が肥大化しにくく，中盤で時間を過剰に消費しにくくなったことが，図 3 で中盤以降にも十分な時間が残されていることがわかる．また，図 5 から，AM 重視の戦略もリスキームーブの割合は変更なしとあまり変わっていないことがわかる．RM 重視の選択でも，リスキームーブの選択自体は増えているものの，それが有効に働く場合とそうでない場合のどちらもあった．

このように適切な範囲で AM の探索を切り上げ，RM の探索に時間を配分することは，計算能力が限られている場合，このエージェントの性能向上にはたらく可能性が高いことがわかった．AM の計算時間を大きく減らしても，RM のモンテカルロ木探索によって精度を保てている．評価値は大きく変わることなく，消費時間の削減が達成できた．

動的重み変更について，勝率はすべての設定で向上した．ただし，その差は小さく，誤差の範囲である可能性がある，しかし，図 7 から，重み変更によって，選択される手の傾向は変化していることがわかる．RM の重みの終盤の値が最も高くなる積極的な戦略で，終盤にリスキームーブが選択される割合と，それがポジティブにはたらく割合はともに高かった．これは，終盤に RM が評価する手を積極的に選択することに戦略的価値があることを示していると考える．

8. おわりに

本研究では，RBC を題材とし，相手の不確実性を利用する手法が動的調整によってどのような場面で有効であるのか特定することを目的として，エージェント Zubat を改良した．特に先行研究では固定的であった計算時間配分と評価値の重みづけを動的に最適化する 2 つの手法を提案した．動的時間配分では，モジュール間の時間配分を工夫することで，少ない計算時間で精度を保つことに成功した．MM 重み動的調整では，ゲームフェーズに応じて重みγを調整することで，リスクテイカー戦略の効果を最大化するようにした．これらは RBC に限らず，他の不完全情報ゲームにおける戦略最適化にも応用可能であると考えられる．

本研究の実験には改善すべき点がある．第 1 に，実装の最適化が必要である．議論で述べたように，現状の実装は先行研究の Zubat に勝率が追い付いていない．これは実行環境の性能差だけでなく，実装の効率性にも原因がある可能性がある．特に AM の盤面評価処理に計算時間が過剰に消費されているため，頻出する盤面の事前評価を行うなど計算効率を改善する余地がある．

第 2 に，実験規模の拡大が求められる．本研究では各パラメータ設定につき 30 回の対戦を実施したが，統計的に有意な差を検出するには不十分である可能性がある．特に動的重み調整の効果は小さく，より多くのサンプル数での検証が必要である．

第 3 に，パラメータ最適化の自動化を検討すべきである．本研究では手動でパラメータを設定し評価を繰り返したが，他の最適化手法を取り入れ対戦を自動化することで，より適切なパラメータを設定可能であると考える．特に第 1 層の時間配分係数については，本研究で提案した設定が十分に機能しなかったため，より体系的なアプローチが必要である．

第 4 に，動的調整アルゴリズムの改善が求められる．本研究では序盤・中盤・終盤の 3 段階に固定的に分類したが，実際のゲーム状況はより連続的に変化する．盤面の複雑さや残り時間などの要因を考慮した調整関数を導入することで，より柔軟な最適化が期待できる．本研究では，そのような複雑な関数は実装に至らなかった．

第5に，SMの計算時間に検討の余地がある．本研究ではモジュール間の時間配分について，SM は相手ターン中にも計算が可能であり持ち時間への影響度が低いため，AM と RM に絞って検討した．しかし，相手がすぐに応手を決めた場合や，相手の手によって駒が失われたりした場合，自分のターンに SM の計算が発生する場合がある．このときの時間配分も，検討する位置の優先順位をつけるなどの実装によって最適化できる可能性がある．

このような課題を解決することで，相手の不確実性を利用する手法をさらに確たるものにすることが可能である．相手の持っている情報を適切に推測し利用することで，他の不完全情報ゲームにも応用できる可能性がある．

文　献

[1] R. Gardner, C. Lowman, C. Richardson, A. Llorens, J. Markowitz, N. Drenkow, A. Newman, G. Clark, G. Perrotta and R. Perrotta, "The first international competition in machine reconnaissance blind chess," *Proc. NeurIPS 2019 Competition and Demonstration Track,* pp. 121-130, 2020.

[2] T. Bertram, J. Fürnkranz and M. Müller, "Supervised and Reinforcement Learning from Observations in Reconnaissance Blind Chess," *Proc. 2022 IEEE Conference on Games (CoG),* pp. 608-611, 2022.

[3] T. Highley, B. Funk and L. Okin, "Dealing with uncertainty: A piecewise grid agent for reconnaissance blind chess," *Journal of Computing Sciences in Colleges, vol. 35, no. 8,* pp. 156-165, 2020.

[4] J. Czupyt, M. Małkiński and J. Mańdziuk, "Capitalizing on the Opponent's Uncertainty in Reconnaissance Blind Chess," *Proc. IEEE Congress on Evolutionary Computation (CEC),* pp. 1-10, 2024.