# コンピュータブリッジにおける並列処理

小田和 友仁　上原 貴夫
東京工科大学

**概要**

不完全情報ゲームであるブリッジでは，相手の手札を推測し想定したさまざまな場合（ワールド）においてゲーム木探索をおこなうために時間がかかる．そこで，各ワールドを1台のコンピュータに割りつけ，複数のコンピュータで並列に処理する方法を提案する．ひとつのワールドでは完全情報ゲームとしてのゲーム木探索ができ，そのワールドは1台のコンピュータ内にあるので，αβ法やトランスポジションテーブルといった高速化の手法が有効に適用できる．しかし，探索の深度を一定としたαβ法を適用すると，カットオフの適用率によりワールドごとの処理時間にばらつきが生じ，遅いコンピュータを待つ時間が増加する．本研究では反復深化法を適用し，一定時間で探索を打ち切ることによりこの問題を解決した．

# Parallel processing of computer bridge

Tomohito Otawa　Takao Uehara
Tokyo University of Technology

**Abstract**

The game tree search is a time consuming task in case of Bridge because Bridge is an imperfect information game and searches must be performed in various worlds generated by guessing opponents' hands. We propose a parallel processing method assigning each world to a computer. Since game tree search can be carried out as a perfect information game in a world and the world is in a computer, the speed up technique, such as alpha-beta algorithm and a transposition table, is effectively applied. When the alpha-beta algorithm with a given depth is applied to search in each world, the processing time of each computer depends on the rate of cutoff, and the waiting time of computers increases. We solve this problem by applying the iterative deepening method within a limited time.

## 1. はじめに

不完全情報ゲームであるブリッジは推測した相手の手札によってさまざまな場合を想定した先読みを必要とするため，完全情報ゲームに比べ探索量が格段に多い．また相手を騙すなど，不完全情報ゲームに特有のプレイテクニックをプログラムで再現しようとすると，さらに多くの処理量を必要とする[1].

人間とコンピュータの対戦を実現するとき，コンピュータの思考時間はルールで定められた時間，あるいは人間が不快に思わない程度の待ち時間に収めなければならない．

本研究室で作製してきたコンピュータブリッジは，思考ルーチンが強化されるたびに探索量が増え，比例して処理時間も膨大なものとなった．より強く人間に近い思考ルーチンを実現していく上で，探索の高速化は欠かせない．しかし，従来の高速化のアルゴリズムを駆使しても，1台のコンピュータによる探索では速度の向上に限界があるため，さらなる高速化の手法が望まれている．

## 2. 目的

不完全情報ゲームであるブリッジのゲーム木探索を，複数のコンピュータで分散し並列におこなうことで高速化を図る．

これにより高度なプレイの可能とするコンピュータブリッジの実現を目指す．

## 3. 完全情報ゲームにおける探索の高速化

完全情報ゲームにおけるゲーム木探索の高速化は，基本的に枝狩りによって実現する．枝狩りを主体とする従来の高速化アルゴリズムには，αβ法やトランスポジションテーブルなどがあげられる．枝狩りにより無駄な探索は省けても，それ以外の枝を狩ってしまうと結果の信頼性が損なわれる．そのため，枝狩りによる速度の向上には限界がある．

## 4. 完全情報ゲームの並列ゲーム木探索

完全情報ゲームのゲーム木を単純に分散処理するならば，ゲーム木のルートと，ルートから伸びる枝とで分轄する方法が　考えられる．各クライアントの探索範囲にのみαβ法を適用すると，ゲーム木全体を範囲とした場合に比べカットオフの効率が減少する．ゲーム木全体をαβ法適用範囲とするには，ネットワークを通して共通のメモリ空間を用意するか，あるいは頻繁な通信が必要となる．

## 5. 不完全情報ゲームのモデル

不完全情報ゲームでは一部の情報しか得られないため，単純にはゲーム木を探索できない．そこで，モンテカルロシミュレーションを適用するのが一般的である[3]．

可視情報から不可視情報を推測し，多数の実現可能な世界をランダムに生成する．それぞれの世界においてゲーム木を生成し探索する．

最終的にリードするカードを選択するには，すべての世界の探索結果を参考にし，不完全情報ゲームとして良さそうなカードを選択する．

このように，モンテカルロシミュレーションによる不完全情報ゲームのモデルは，いくつかの世界とその世界ごとにひとつのゲーム木を持つ形をとる．（図 1）

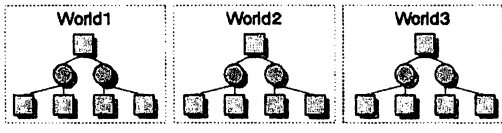


図 1　不完全情報ゲームのモデル

## 6. 不完全情報ゲームの並列ゲーム木探索

不完全情報ゲームの並列処理ではモンテカルロシミュレーションによる不完全情報ゲームのモデルを利用する．モデルの各世界をクライアントに分担させることで，クライアントは完全情報ゲームとして探索をおこなえるので，従来の高速化アルゴリズムを適用できる．

最初の 1 手のみを不完全情報ゲームとして選択することで，より人間的な思考に近づけることができる．クライアントではゲーム木全体を一貫して探索するが，サーバに結果として返すのは，ルートから伸びる枝（次の手の候補）とその枝の評価値をまとめたレポートとする．サーバでは各クライアントのレポートをひとつにまとめ，その中から評価値の高いものを最良の手として選択する．

### 6.1. クライアントの処理時間

クライアントの探索木にαβ法やトランスポジションテーブルなどを適用すると，探索木ごとにカットオフの発生率が異なるため，処理時間にばらつきが生じる．

ワールドの処理時間のばらつきはオーバヘッドを引き起こし，並列処理の効率を下げる．

### 6.2. クライアントのばらつきの解消

クライアントのばらつきの対処法として二つの方法が考えられる．反復深化法とスケジューリングである．これらの両立も可能ではあるが，効率が上がるとは限らない．スケジューリングでは 1 台のクライアントが受け持つワールド数が多いほど効率が上がるが，ワールドの処理時間との兼ね合いであまり増やすわけにもいかない。そのため、本研究ではまず反復深化法による解決を選択した。

### 6.3. 反復深化法による解決

反復深化法は深度 1，2，3...と深くしながら繰り返し MinMax 探索していく方法である．深度 n における探索順を深度 n-1 での評価値をもとに並べ替えることでαβ法におけるカットオフの適用率を上げることができる．

反復深化法では探索ごとに探索結果が得られている．これを利用し一定時間で探索を切り上げることで，ワールドの処理時間を平均化することができる．

深度 n で探索を打ち切ると，深度 n-1 での完全な探索結果がえられる．また深度 n での探索で最終結果は得られずとも，ある程度次手候補の評価値が得られている場合もある．

最終的に不完全情報ゲームとして選択する場合，次手の候補と評価値を必要とするので，次手候補の最深探索での評価値を結果として返す．つまり次手候補について深度 n 探索で評価値が得られているならばこれを返し，そう

でない場合は深度 n-1 探索での評価値を返す．図 2 に例をしめす．まず，深度 1 の探索をすべて終了した．次に深度 1 で得られた評価値から探索順を降順に並び替え，深度 2 の探索を開始する．節 B の探索を終え，節 C の探索の途中で，指定した時間が過ぎたとする．このとき，深度 2 における次手の最終決定はなされないが，節 B における評価値は得られている．ここで採用する次手の候補の評価値は，A および C については深度 1 のもの，B については深度 2 のものとする．

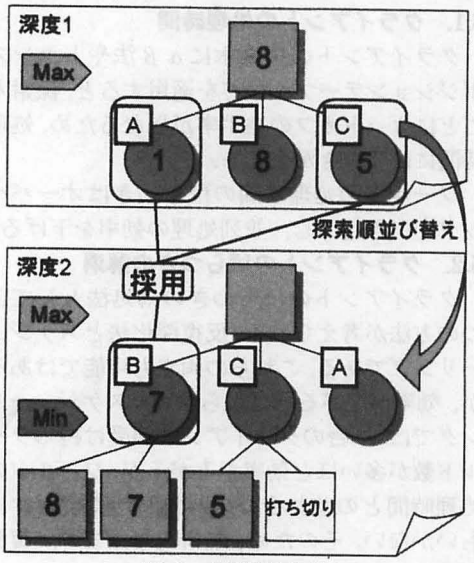


図 2　反復深化法

## 7. 開発言語

開発言語には ECLiPSe をもちいる．ECLiPSe は制約論理プログラミング言語であり，制約条件による問題表現と，述語論理形式でのプログラムを可能とする言語である[4].

ECLiPSe は多数のライブラリを備えており，ソケット機能も提供しているので，クライアント／サーバ間の通信にはこれを利用する．

## 8. 実験方法

反復深化法を適用し時間で探索を切り上げるようにしたコンピュータブリッジについて，クライアントの処理時間が平均化されていることと，プレイの正当性を調べるため，改良したコンピュータブリッジと未改良のものとをそれぞれ実験で比較する．

実験環境として，クライアント用に同じ性能のコンピュータ 8 台に，サーバ用にコンピュータ 1 台を用意する．それぞれを 100BASE の LAN に接続する．

### 8.1. 処理時間の平均化の確認

ばらつきが解消されていることを確認する．オープニングリードについてワールドを多数生成しクライアントに探索させ，1 ワールド毎の処理時間を計測する．

### 8.2. プレイの妥当性と処理時間

プレイの妥当性を確かめるために，反復深化法を適用したコンピュータブリッジをゲームが終了するまで（13 トリック）4 人分を探索しプレイの履歴を検証する．

クライアントを 8 台としたときの，ワールド数とリミット時間の組み合わせによるクライアントの予想処理時間を表 1 に示す．

コンピュータブリッジにおいてコンピュータが手を決定するための思考時間は多くても 10 秒程度が限界である．これ以上の時間がかかると，公式のブリッジ大会が定めたルールの枠を超える可能性がある．また，人間が不快に感じることだろう．そこで，表 3 において予想処理時間が 8 となる組み合わせで実験をおこない，プレイの履歴を比較する．また，リミット時間の増減によりプレイにどの程度の妥当性が得られるかを調べるため，ワールド数を 8 に固定しリミット時間を変えて実験をおこない，プレイの履歴を比較する．

表 1　クライアントの予想処理時間（秒）

| | | ワールド数 | | |
|---|---|---|---|---|
| | | 8 | 16 | 32 |
| リミット時間（秒） | 2 | **2** | 4 | **8** |
| | 4 | **4** | **8** | 16 |
| | 8 | **8** | 16 | 32 |
| | 16 | **16** | 32 | 64 |

## 9. 実験結果

### 9.1. 処理時間の平均化の確認の実験結果

ワールドごとの処理時間にばらつきが見られず，ほぼ正確にリミット時間で打ち切っていることを確認した．

### 9.2. プレイの妥当性と処理時間の実験結果

表 2，表 3 はそれぞれディール A，ディール B におけるプレイ開始から終了までの総処理時間を 8.2.で述べた組み合わせの条件で計測した結果である．

表2 ディールAの総処理時間（秒）

| リミット時間（秒） | ワールド数 8 | 16 | 32 |
|---|---|---|---|
| 2 | 139.0 | | 523.3 |
| 4 | 197.5 | 373.7 | |
| 8 | 288.9 | | |
| 16 | 506.8 | | |

表3 ディールBの総処理時間（秒）

| リミット時間（秒） | ワールド数 8 | 16 | 32 |
|---|---|---|---|
| 2 | 133.9 | | 519.8 |
| 4 | 192.0 | 356.8 | |
| 8 | 277.2 | | |
| 16 | 477.2 | | |

一手分の予想時間が 8 秒となる組み合わせでは，表 5.1，表 5.2 いずれの場合も総処理時間に大きな違いが生じた．ワールド数が多く，リミット時間が短い設定の方が処理時間が長いという結果となった．

リミット時間の増減はクライアントの処理量に影響を及ぼすが,サーバの処理量にはほぼ影響しない．リミット時間が短いとクライアントの処理量が小さく，サーバの処理量の比重が大きくなってしまうため，このような結果になったと考える．

また，リミット時間が長いと，ゲーム後半ではカード枚数に頭打ちされ,リミット時間に満たないうちに探索を終了することも原因と考える．

ワールド数 8 固定での結果は，この頭打ちによる影響が大きいといえる．

以上から，リミット時間は長いほど並列処理による効率が得られ，リミット時間が十分に長ければ，1 台のクライアントが担当するワールド数が多いほど効率的であるといえる．

ただし思考時間には限界があるので，今後実験を重ね，最適なワールド数とリミット時間の組み合わせを調整していく必要がある．

## 10. 今後の計画

8.の実験をさらに重ね，プレイの妥当性を確認しながらワールド数とリミット時間の調整を進める．

### 10.1. C 言語への移植

ECLiPSe はインタプリタ方式の言語である．コンパイラ方式はインタプリタより高速なプログラムの実行を可能とするため，コンパイル方式を採用している C 言語への移行を進めている．ただし，オークション部のビッドに関しては制約条件をもちいているので，ECLiPSe によるプログラムを流用する．よって ECLiPSe のプログラムと C 言語のプログラムの接続が必要となるため，ソケットをもちいる方向で方法を模索している．

C 言語はマルチスレッド機能も提供しているため，C 言語版のコンピュータブリッジが完成しだい，スケジューリングを実装する．

### 10.2. 不完全情報ゲームとしての処理

サーバとクライアントに関して不完全情報ゲームとしての処理を強化することで，上級者のプレイの実現を目指す．

#### 10.2.1. サーバの改良

最初の 1 枚だけの選択では問題（Two Way Finess）があるので，少なくとも 1 トリック（4 枚）まで処理する．

#### 10.2.2. クライアントの改良

エージェントモデルに基づくアルゴリズム（自己の推論した世界と他者の推論した世界をペアにして探索）を実装することで，ディセプティブプレイを実現する[1].

### 10.3. 数 100 台規模の実験

上級者のプレイと比較評価する．

## 11. 参考文献


[1] 小林 紀之・上原 貴夫：コンピュータブリッジによるディセプティブプレイ，情報処理学会論文誌 Vol.43 No.10 pp.3056 – 3063（2002）

[2] 黒川 昌夫 著「ブリッジ上達法」有紀書房（1998）

[3] Ginsberg, M.L : GIB: Imperfect information in acomputationally challenging game, Journal of Artificial Intelligence Research (2001)

[4] IC-Parc「The ECLiPSe Constraint Logic Programming System」 http://www-icparc.doc.ic.ac.uk/eclipse/