2003－GI－10　(4)
2003／8／4

不完全情報ゲームにおける並列処理

Parallel processing of incomplete information game

小田和 友仁†　　**上原 貴夫**†
Tomohito Otawa　　Takao Uehara

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.

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

チェスや将棋など, ゲーム上のすべての情報を把握できるゲームを完全情報ゲームという. 一方, トランプゲームのブリッジやポーカーなど，一部の情報（相手の手札など）が取得できないゲームを不完全情報ゲームという.

近年, ゲーム研究の範囲は完全情報ゲームから, より複雑な探索を必要とする不完全情報ゲームへと拡大しつつある.

不完全情報ゲームでは推測した相手の手札などについて, 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, 処理時間はさらに長くなる[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.

†東京工科大学
Tokyo University of Technology

本研究室で製作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. 今後, 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. コントラクトブリッジ

ブリッジはテーブルを囲み 4 人でプレイするカードゲームである[2]. 対面のプレイヤ同士でペアを組み競い合う. ジョーカーを抜いて, 4 スート[1]×13 枚で 52 枚のカードをもちいる. カードは 4 人に 13 枚ずつディール[2]する.

他のプレイヤのハンド[3]が見えない状態で, オークションと呼ばれる競り合いをおこなった後, プレイを開始する.

通常はこのオークションとプレイを数回繰り返し, 最終的な勝敗を決定する.

3.1. オークション

オークションでは 4 人のプレイヤが順に, プレイにおいて取る見込みのあるトリック[4]数とトランプ[5]かノートランプ[6]をビッド[7]する.

ビッドではトリック数 7 以上を宣言する必要があり, トリック数が多いほど順位が高い. 最高のトリック数は 13 である. 同じトリック数ではノートランプがトランプより高いビッドであり, トランプのスートは S（スペード）＞H（ハート）＞D（ダイヤ）＞C（クラブ）の順に高い.

ビッドではトリック数とトランプの有無をまとめて呼称する. 例えばクラブで 7 トリックの場合 1C（ワンクラブ）, ノートランプで 9 トリックの場合 3N（スリーノートランプ）と宣言する.

また, ビッドにはダブルとリダブルがある. ダブルは前のプレイヤと同条件で勝利時の得点を倍とする（ただし勝利できなかったときには倍以上の得点を取られる）ビッドである. リダブルは, ダブルをビッドされたときのみ宣言でき, 同条件でさらに得点を倍にする（ただし勝利できなかったときにはさらに倍以上の得点を取られる）ことができる.

次のプレイヤはより高いビッドかパスを宣言しなければならない. あるビッドに対し他の 3 人がパスを宣言したとき, そのビッドが採用されコントラクト[8]となる.

コントラクトを宣言したプレイヤが属するペアのうち, コントラクトのスートを最初にビッドしたプレイヤをディクレアラ[9]と呼び, ディクレアラのパートナーをダミー[10]と呼ぶ. もう一方のペアはディフェンダ[11]と呼ぶ.

コントラクトが決定すると, プレイの開始となる.

3.2. プレイ

ディクレアラの左隣のプレイヤがリード[12]することでプレイを開始する. このリードをオープニングリード[13]という.

---

1 スート　カードのマーク（スペード, ハート, ダイヤ, クラブ）
2 ディール　カードを分配すること
3 ハンド　ディールされた手元のカード
4 トリック　プレイ中, 4 人の中で一番強いカードを出して集めた 1 組（3.2.参照）
5 トランプ　切り札となるスート
6 ノートランプ　切り札としてスートを指定しない
7 ビッド　取る見込みのあるトリック数とトランプかノートランプを宣言すること
8 コントラクト　取ると宣言したトリック数とトランプの有無のペア
9 ディクレアラ　コントラクトを宣言したプレイヤ
10 ダミー　ディクレアラのパートナー
11 ディフェンダ　コントラクト達成を阻止するペア
12 リード　ハンドから 1 枚テーブルに出すこと
13 オープニングリード　プレイ開始のためのリード

オープニングリード後，ダミーのハンドはテーブルに表にして広げ，ダミーのリードはディクレアラがおこなう．

リードは時計回りにおこなう．二人目以降のリードでは，最初に出されたカードと同じスートのカードを出さなければならないが，ハンドに同じスートが存在しない場合のみ別スートのカードを出しても良い．

4 人分のカードがテーブルに出揃ったところで，カードの強さを比較する．カードはトランプのスートがトランプ以外のスートよりも強く，同スートでは A＞K＞Q＞J＞10＞…＞2 の順に強い．

一番強いカードを出したプレイヤがその回のウィナーとなりトリックを得る．ウィナーは次回の最初のリードをおこなう．

以上の勝負を 13 回繰り返し，勝敗を決める．ディクレアラ側がコントラクトを達成したならばディクレアラ側の勝利となり，コントラクトとコントラクトを上回ったトリック数から計算される得点をディクレアラ側が得る．逆に達成できなかった場合，ディフェンダの勝利となり，コントラクトとコントラクトに足りないトリック数から計算される得点をディフェンダが得る．

4. 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[3]やトランスポジションテーブルなどがあげられる[4]．枝狩りにより無駄な探索や重複する探索は省けても，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．

ハードウェアの性能向上による処理速度の向上にも技術的な，あるいはコスト的な限界が存在する．限られたハードウェア性能の枠内で高速なシステムを組む場合，並列処理による実現がもっとも効果を上げやすい．

5. 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，図 1 のように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．これにはソフトウェアの対応はもとより，ハードウェアの環境整備が望まれる．

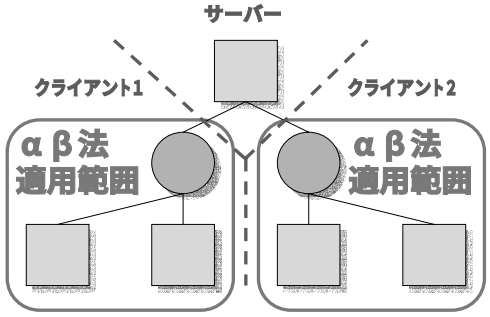

図 1　完全情報ゲームの並列ゲーム木探索

6. 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[5].

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 2）

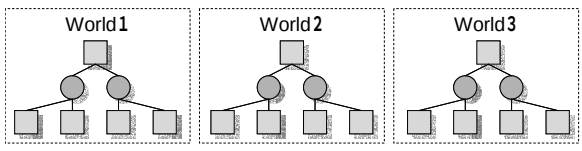

図 2　不完全情報ゲームのモデル

7. 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, αβ法やトランスポジションテーブルといった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.

クライアント／サーバ間の通信は, 探索開始時のワールド送信と終了時の結果送信だけですため, 通信時間によるオーバヘッドはあまり全体に影響を及ぼさない.

7.1. 実現方法

あらかじめクライアントプログラムを実行し, 受信待ち状態にしておく. サーバは可視の情報をもとに, 実現可能な世界をランダムに生成し, 各クライアントに送信する. すべて送信したのち, サーバは結果の受け取り待ちに移る. クライアントは世界を受け取ると, 完全情報ゲームとしてゲーム木探索をおこない, 次手の候補とその評価値をレポートとしてまとめサーバに送信する. サーバは各クライアントから送られてきたレポートを統合し, 最善手を決定する.（図 3）

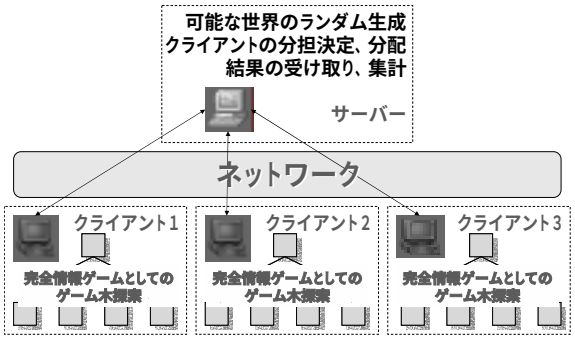

図 3　不完全情報ゲームの並列ゲーム木探索

7.2. 使用言語

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[6].

本研究室のコンピュータブリッジでは, ビッド部を制約条件で表現している. また, 本研究の対象であるプレイ部ではワールド生成のみ, 制約条件をもちいて表現しており, ワールド毎の探索はゲーム木探索を基本としている.

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.

7.3. 並列探索の台数効果

表 1 は並列処理を施したコンピュータブリッジにおいてオープニングリードにおける探索を, 条件を変えてそれぞれ 5 回測定した処理時間の平均値である.

探索の深さ, ワールド数に関わらず, クライアントによる台数効果はある程度は得られていることが確認できる.

クライアントを 2 倍 4 倍と増やしても処理時間が 1/2, 1/4 とならない. これは, クライアント台数の増加によりクライアント 1 台の担当するワールド数が減少したため, ワールドの処理時間のばらつきによる処理待ちの影響が顕著に現れるためである. ワールド数 12 よりもワールド数 4 の方が, 台数効果が低いことからも処理待ちの影響が確認できる.

表 1　平均処理時間（秒）

探索の深さ		ワールド数	クライアント台数 1	2	4
8		4	1.3	1.0	0.8
		12	3.9	2.5	1.5
16		4	61.1	46.4	39.1
		12	328.6	184.4	119.1

7.4. クライアントの処理時間

クライアントの探索木にα β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.

表 1 はα β法を適用したコンピュータブリッジのクライアントに, 同じ制約条件のもとでランダムに生成した 20 個のワールドをひとつずつ深度 16 で探索させ，オープニングリードの決定に要する処理時間を測定した結果である．最短で 2.2 秒，最長で 78.0 秒と，大きな開きがある.

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.

表 2　1 ワールドの処理時間

番号	処理時間（秒）	番号	処理時間（秒）
1	**2.2**	11	7.4
2	7.4	12	38.0
3	11.8	13	13.7
4	38.7	14	6.1
5	38.0	15	33.8
6	20.5	16	46.2
7	4.9	17	11.8
8	25.4	18	6.9
9	**78.0**	19	6.8
10	53.8	20	8.7

7.5. 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ふたつの方法が考えられる. ひとつは反復深化法, もうひとつはスケジューリングである. 前者はそもそものばらつきを無くす考え方で, 後者はばらつきを前提に処理順を工夫することで全体的な平均化を果たす考え方である. 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.

7.5.1. 反復深化法による解決

反復深化法とは深度 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 [4]．深度 n における探索順を深度 n-1 での評価値の降順とすることでα β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 4 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.

このアルゴリズムをプログラムに実装するには, 次手の候補とその評価値を保存するテーブルを用意する.

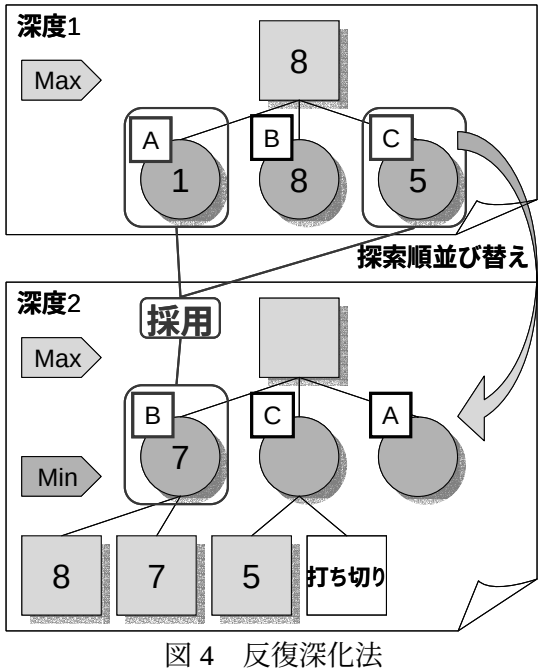

図 4　反復深化法

7.5.2. スケジューリングによる解決

並列処理におけるスケジューリングとは, クライアントの処理時間のばらつきによる処理待ちを回避するため, クライアントに割り振るタスクの順番を計画することである. ひとつのクライアントが受け持つタスク数が多いほど, 組み合わせパターンの選択枠が広がり効率が上がる.

理想的には, 処理させたいタスクのクライアントでの処理時間がわかっていれば, あらかじめ全クライアントの総処理時間が平均的になるよう計画できる.

現実的には, クライアントでの処理時間は予想に留まるか, もしくはまったくわからない. この場合, すべてのクライアントにタスクを割り与え処理の終了を監視し, 終わりしだい次のタスクを与える方法が適している.

スケジューリングをおこなうには, 全クライアントの処理を並列に監視しておく必要がある. そのため, サーバの処理をマルチスレッドにする必要がある.

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. この結果を, 同条件で以前おこなった反復深化法未適用のコンピュータブリッジでの実験結果（表 2）と照らし合わせ, ワールド毎の処理時間が平均化されていることを確認する.

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 3 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 となる組み合わせが理想に近いと仮定し, この組み合わせで実験をおこない, プレイの履歴を比較する. すなわち, 次の組み合わせである.

- ワールド数 8　　リミット時間 8 秒
- ワールド数 16　　リミット時間 4 秒
- ワールド数 32　　リミット時間 32 秒

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. すなわち, 次の組み合わせとなる.

- ワールド数 8　　リミット時間 2 秒
- ワールド数 8　　リミット時間 4 秒
- ワールド数 8　　リミット時間 8 秒
- ワールド数 8　　リミット時間 16 秒

表 3　クライアントの予想処理時間（秒）

リミット時間（秒） \ ワールド数	8	16	32
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

表 4 は反復深化法を適用したコンピュータブリッジのクライアントに, 同じ制約条件のもとでランダムに生成した 20 個のワールドをひとつずつリミット時間 8 秒で探索させ, オープニングリードの決定に要する処理時間を測定した結果である.

番号 1〜20 の処理時間に少数点以下第 1 位までの精度で数値の違いがなく, ほぼ正確にリミット時間で打ち切っていることがわかる.

表 4　1 ワールドの処理時間

番号	処理時間（秒）	番号	処理時間（秒）
1	8.0	11	8.0
2	8.0	12	8.0
3	8.0	13	8.0
4	8.0	14	8.0
5	8.0	15	8.0
6	8.0	16	8.0
7	8.0	17	8.0
8	8.0	18	8.0
9	8.0	19	8.0
10	8.0	20	8.0

9.2. プレイの妥当性と処理時間の実験結果

プレイの妥当性に関してはさまざまなディールでの実験を必要とするため, 現在データの収集を進めている.

表 5.1, 表 5.2 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.

また, 表 6 は 52 枚分の探索時間として表 3 の時間を 52 倍したものである. ゲーム終盤の探索量はハンドのカード数に頭打ちされ徐々に小さくなるため, クライアントの処理時間も短くなる. また, クライアントの処理時間とは別にサーバの処理時間, 通信のオーバヘッドが加わるため, 表 6 は正確に予想されたものではなく, 考察の基準としてのみ参照するものである.

表 5.1　ディール A の総処理時間（秒）

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

表 5.2　ディール B の総処理時間（秒）

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

表 6　基準の総処理時間（秒）

		ワールド数		
		8	16	32
リミット時間（秒）	2	104		416
	4	208	416	
	8	416		
	16	830		

表 6 と表 5 を比較すると, リミット時間 2 秒ではディール A･B の処理時間が基準を上回っているのに対し, リミット時間 4 以上ではディール A･B の処理時間が基準の処理時間を下回っている.

リミット時間 2 秒程度ではクライアントの処理量は非常に小さく, サーバの処理量と通信のオーバヘッドの比重が大きいために, 基準を上回ったものと考える. さらに, ワールド数 32 ではワールド生成にかかる時間も多くなり, この傾向が顕著になる.

リミット時間 4 秒以上では, 逆にクライアントの処理量の比重が大きくなったために基準を下回る結果が得られた. リミット時間を増やしてもサーバの処理量はほとんど増加せず, 通信のオーバヘッドがわずかに増す程度で, クライアント処理の増加量に比べると非常に少ない.

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. まとめ

完全情報ゲームの分野で研究されてきた従来の高速化アルゴリズムを活かしながら並列処理の台数効果も上げることのできる不完全情報ゲームの並列ゲーム木探索の実装法を立案した.

12. 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] Leon Sterling, Ehud Shapiro 著
松田 俊夫 訳「Prolog の技芸」共立出版 (1988)

[4] 松原 仁・竹内 郁雄 編
「bit 別冊 ゲームプログラミング」
共立出版(1997)

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

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