6L-7

コンピュータブリッジにおける並列ゲーム木探索
MPI による実装

小田和 友仁 †
東京工科大学 †

上原 貴夫 ‡
東京工科大学 ‡

1 はじめに

コントラクトブリッジ（以下ブリッジ）は 4 人でプレイするカードゲームである．他人に見えないように持つ手札があるため，不完全情報ゲームに分類される．不完全情報ゲームは，完全情報ゲームと同じようにゲーム木を構成することはできない．

最近のコンピュータブリッジにおける最前手決定法はモンテカルロ法に基づくアルゴリズムが一般的である．未知の情報に対して複数の仮説を立て，各仮説において完全情報ゲームとして探索し，結果を集計して最善手を決定する．このアルゴリズムは複数の完全情報ゲーム木を探索しなければならないため探索量が多く，処理時間がかかる．

本稿ではこのアルゴリズムに基づき，高い台数効果を得ることができる並列処理の実装法を述べる．

2 コンピュータブリッジ

我々はブリッジのプレイヤを，知識と仮説推論機構をもつエージェントとしてモデル化した．エージェントはビッドやプレイの経過などの局面情報を観察し，知識と照らし合わせることで推論をおこない，結果を制約条件にまとめる．この制約条件に矛盾しない手札の仮説を多数生成し，モンテカルロ法の原理に従い最前手を決定する [1]．本研究では手札の仮説をワールドと呼び，この一連の処理をワールド生成と呼ぶ（図 1）．

制約条件を扱うため，制約論理プログラミング言語である ECL^{i}PS^e[3] を中心に実装を進めてきた．高速化のため ECL^{i}PS^e による並列化を果たした．また，ECL^{i}PS^e はインタプリタ言語であり実行速度が遅いため，C 言語へ移植することで高速化を図ってきた．本研究では更なる高速化のため C 言語により並列処理を実装する．

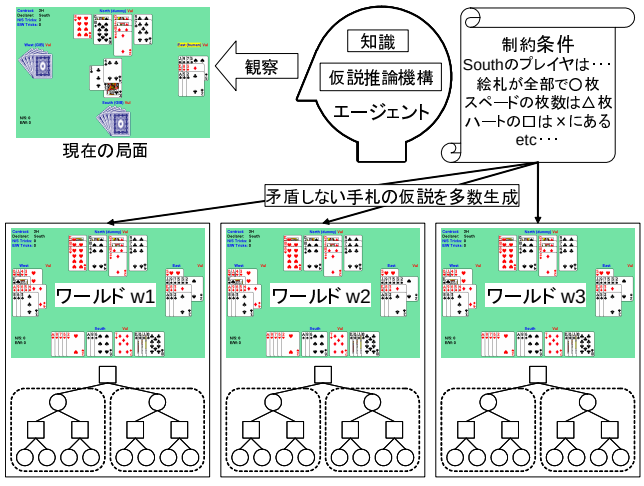

図 1 ワールド生成

Parallel Game Tree Search for Computer Bridge with MPI
† Tomohito Otawa, Tokyo University of Technology
‡ Takao Uehara, Tokyo University of Technology

2.1 モンテカルロ法に基づくアルゴリズム

見えていないカードの可能な分布（ワールド）を多数生成し，各世界でダブルダミーブリッジ（4 人の手札を見た状態でプレイ）の MinMax 値を求め，候補の中から全体として良さそうな行動を選ぶ方法が何人かの研究者により提案された．Ginsberg の GIB もこの方法をもちいている [2]．Ginsberg は出しうるカード候補の集合を M としたとき，つぎのようにして最前手を決定している．

[Step1] それまでのビッドおよびプレイと矛盾しないようにカードをくばり，ディールの集合 D を作る

[Step2] 各ディール $d \in D$ ごとに，各行動 $m \in M$ を選択した場合どのような結果になるかをダブルダミーで評価し，スコア $s(m,d)$ を計算する

[Step3] $\sum_d s(m,d)$ が最大となるような行動 m を選ぶ

2.2 不完全情報ゲームのモデル

2.1 のアルゴリズムによる不完全情報ゲームのモデルは，図 1 のようにいくつかのワールドとワールドごとに 1 つのゲーム木を持つ形をとる．各ワールドのゲーム木をもちいて完全情報ゲームとして探索できるため，完全情報ゲームの分野で培われてきたα β法などの高速化アルゴリズムを実装できる．

2.3 α β法の適用範囲

2.1 の [Step2] における候補ごとの評価値 $s(m,d)$ はかならず結果を返す必要があるため，ルートでのα，β値の更新は危険である．そこで，ルートの子ノードを頂点とする個々の部分木をそれぞれα β法の適用範囲とする．図 1 のゲーム木にα β法の適用範囲を破線で示した．

3 不完全情報ゲームの並列処理

2.2 で述べたモデルの特性を活かした並列処理の実装法を提案する．システムの構成を図 2 に示す．

ワールドは完全情報ゲームとして探索するためのデータが独立しているため，タスクの単位として適する．さらに 2.3 で述べたα β法の適用範囲はワールド内に候補数だけ存在するが，それぞれのタスク並列性は高い．また，タスク数が多く個々の処理時間が短い方が，通信オーバヘッドが増えるものの動的スケジュールによりタスク分配が最適化される可能性が高まる．そのため，α β法の適用範囲，つまりルートの子ノードを頂点とする部分木はタスクの単位として最適である．本研究ではこの部分木をタスクの単位として採用した．

タスクの生成はワールドの生成を含む．ワールドの生成は 2 章で述べたとおり ECL^{i}PS^e により制約条件をもちいて実現している．このワールド生成部は C 言語への移植が非常に難しい．そこで，ECL^{i}PS^e のワールド生成プログラムとソケット通信で連携をとる．

タスクの管理にはタスクプールシステムによる動的スケジューリングをマスター／スレイブモデルとして実装する．図 3 に最前手決定の流れを表したフローチャート

を示す．マスターはタスクの生成，管理と処理結果の集計を担当し，スレイブはタスクの処理に専念する．

並列処理を実装するためにはクラスタの管理や，クラスタ間のメッセージ通信が必要である．これらを実装するために MPI ライブラリである LAM[4] をもちいる．

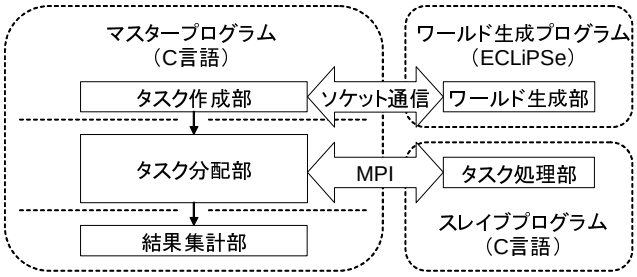

図 2　システム構成

3.1　MPI(Message Passing Interface) ライブラリ

MPI は並列プログラミング用の API 仕様である．仕様に準拠した実装ライブラリに LAM がある [4]．LAM は UNIX ベースの OS のみをサポートし，Windows 系列の OS では動作しない．UNIX のデーモンとして常駐し，これを介して通信するため非常に高速な通信が可能である．

3.2　SPMD(Single Program Multi Data stream)

MPI の提供するプログラミングモデルは，各ノードで異なるプログラムを実行する MPMD(Multi Program Multi Data stream) ではなく，SPMD である．すなわち，すべてのノードがひとつのプログラムを読み込み実行する．ただし，各ノードに割り当てられた識別 ID により処理を分岐することは可能である．

マスター／スレイブモデルを実現するために，この識別 ID を利用する．

3.3　マスタープログラム

マスターは主に 2.1 の [Step1] と [Step3] を担当する．図 3 をもちいてマスタープログラムの流れを説明する．

はじめに 2.1 の [Step1] に従いタスク生成をおこなう．局面の情報をワールド生成プログラムに送信し，ワールド生成の依頼をする．ワールドの受け取り待ちに移り，ワールド生成プログラムが送信してきた結果を受け取る．また，手札から出しうるカードの候補を発生し，ワールドと候補の順列組み合わせをタスクとする．

続いてタスク分配部に移る．マスターはまず，全スレイブに対して 1 つずつタスクを送信する．処理が終了し，結果を送信してきたスレイブに対して，タスクが残っているのであれば次のタスクを渡す．すべてのタスクを送り出し，その結果が帰ってくるまでこれを繰り返す．

スレイブから受け取る結果は候補のカード情報と，そのスコア $s(m,d)$ とする．全タスクの結果がそろったところで，2.1 の [Step3] に従い集計に移る．結果として受け取ったカードの候補ごとにスコアの総和 $\sum_d s(m,d)$ を計算し，最大となるようなカードの候補 m を最善手として決定する．

3.3.1　ワールド生成プログラム

ワールド生成部は ECL^{i}PS^e により実装した．マスターからのワールド生成依頼を受け取ると，送られてきた情報をもとに制約条件をまとめ，ワールドを生成する．生成したワールドを結果としてマスターに送信し，マスターからの依頼待ちへと戻る．

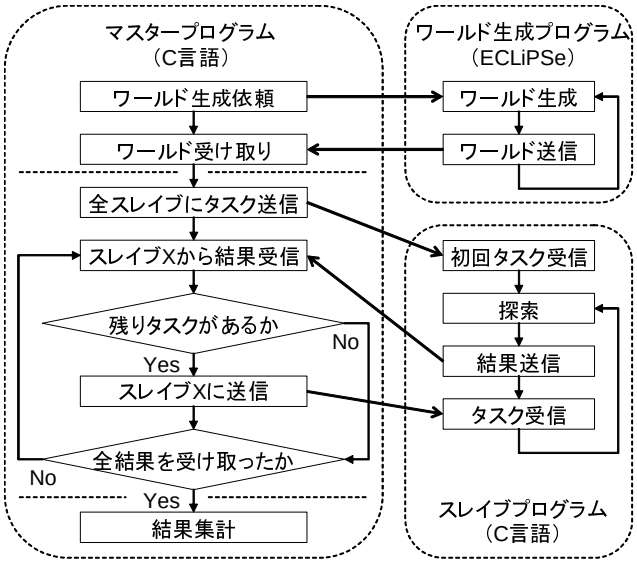

図 3　フローチャート

3.4　スレイブプログラム

スレイブは 2.1 の [Step2] を担当する．マスターから送られてきたタスクのワールドをもとに，プレイの経過からすでに出されたカードを取り除き，現在の局面まで更新する．さらにタスクに付随する候補のカードをプレイした場合のスコアを完全情報ゲームとして探索する．結果として得られたスコア $s(m,d)$ と候補のカード情報をマスターに送信する．

4　最後に

コンピュータブリッジの最前手決定法として一般的にもちいられるモンテカルロ法に基づくアルゴリズムの特徴を活かした並列処理の実装法を述べた．この実装法はモンテカルロ法に基づくアルゴリズムを採用するプログラムに応用可能である．

発表では実験結果を示しこの並列処理の実装法が有効であることを示す．今後はスレイブの α β 探索に加え，パーティションサーチなどの高速化アルゴリズムを実装する．さらに，以前に提案した他者の推論を考慮したアルゴリズム [5] についても並列処理を実装する予定である．

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] Ginsberg M. L., ”GIB: Steps toward an expert-level bridge-playing program”, IJCAI-99 (1999)

[3] IC-Parc, ”The ECLiPSe Constraint Logic Programming System”, `http://www.icparc.ic.ac.uk/eclipse/`

[4] Trustees of Indiana University, ”LAM/MPI Parallel Computing”, `http://www.lam-mpi.org`

[5] 小田和友仁, 上原貴夫, ”コンピュータブリッジにおける新しいアルゴリズムと高速化”, 第 9 回ゲーム・プログラミング ワークショップ 2004, pp.64-70 (2004)