ロビーサービスを用いたゲーム処理の高速化

郡司　直廉、森　秀樹、上原　稔
東洋大学工学部情報工学科

1. はじめに

一般に、チェス、将棋、囲碁、リバーシ（オセロ）などの二人零和有限確定完全情報ゲームの人工知能は、ゲーム木という探索木を用いて先読みを行うが、ゲーム木の探索には先読みの手数や分岐因子が多いほど時間がかかる。そのため、チェスより複雑で、分岐因子も多い将棋や囲碁の人工知能はまだまだ弱いままである。[1][2]

本研究では、ロビーサービス型のゲームサーバとそれに接続した汎用のパーソナルコンピュータを用いて分散処理を行い、ゲーム木の探索速度を向上させる方法を提案する。この方法の特徴は、インターネットに接続された汎用の PC を用いるため、専用コンピュータを用いて分散処理する場合に比べ安価であるという点、ユーザーがゲームで遊んでいる間に CPU パワーを提供してもらうため、ユーザーがほとんど恩恵を得られない現在の各種分散コンピューティングプロジェクトに比べユーザー数を増やしやすいという点である。また、新たに Java 言語でも実装を行ったため、プラットフォームを選ばないという特徴もある。[3]

本論文では、提案システムの設計と実装について議論し、実験を行うことで提案システムの有効性を確認する。

2. 人工知能の仕様

本研究では、分散処理によるゲーム木の探索の高速化を目的とするため、ゲームには比較的単純なリバーシを用いた。[2]

ゲーム木の探索を高速化するためには、枝刈りや評価関数のアルゴリズムの最適化も重要であるが、これらは各ゲームに固有のものもあるため、最適化はゲーム毎に行う必要がある。そのため本研究では、各ゲームに依存する部分に関しては、人工知能の基本的なアルゴリズムとした。具体的には以下の通りである。

- ゲーム木の探索法は、一般的なミニマックス法を用いた深さ優先探索とした。
- 枝刈りのアルゴリズムには、各ゲームに依存しないアルファ・ベータ法を使用した。
- 評価関数は、相手に取られない石（角に置いた石など）に重みをつけて計算するアルゴリズムとした。

The speeding-up of game processing for Lobby Service
Naoyuki Gunji, Hideki Mori, Minoru Uehara
Dept. of Information and Computer Sciences, Toyo University

3. ロビーサービス型ゲームサーバのアーキテクチャ

本システムでは、ゲームサーバを、ユーザー同士の対戦の他にユーザー対コンピュータ（人工知能）の対戦も行えるようにする。

図 1 は、ゲームサーバとクライアントの接続を図示したものである。既存のロビーサービスと異なる点は、各クライアント PC が人工知能の処理を行う専用のスレッド（人工知能ノード）を持ち、人工知能との対戦を行うクライアントは分散処理の中心となる人工知能サーバのスレッドを持つという点である。これらのスレッドがクライアント PC の余っている CPU パワーを用いて人工知能の処理を行う。

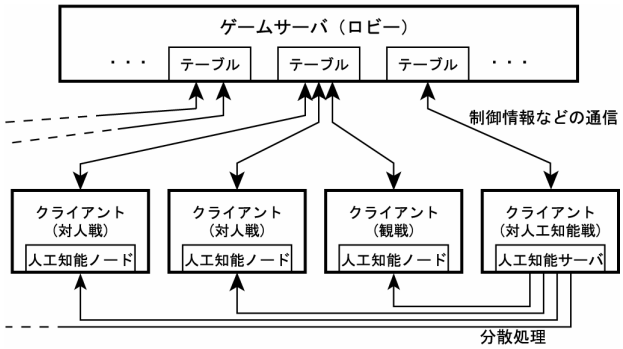

図 1: 接続図

4. 分散処理における探索アルゴリズム

ゲーム木の探索において、木の根に近い部分は人工知能サーバが、その先は複数のノードが分散して探索を行う。図 2 にその模式図を示す。

この方法の特徴は、通信するデータのサイズが小さく（最大 21 バイト）、通信の回数が少ないということである。このため、ネットワークの速度の影響をほとんど受けることが無い。

しかし、分散処理をすることで、本来ならば枝刈りされる節点も一部ではあるが探索を行うため、その分効率が悪くなるという欠点がある。

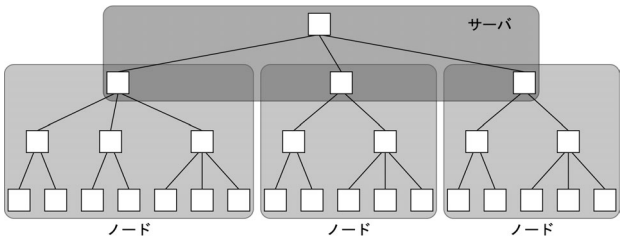

図 2: 分散処理の手法

5. 実験と評価

5.1. 実験の内容

分散処理による探索速度の変化を調べるための実験を行った。実験内容は、すべての PC がゲームサーバに接続し、そのうち 1 台が対人工知能戦、残りが対人戦を行っている場合とした。（この場合、すべての PC が 1 つの人工知能の処理を分散して行うことになる。なお、ここでは、人工知能サーバを動かす PC で、人工知能ノードも同時に動作させている。）この状況で、序盤（4 手後の局面）から終盤（52 手後の局面）までの 5 つの状況において、それぞれ 7 手先までの探索を行い、探索時間を測定した。また、以下の式より探索時間の理論値と分散処理の効率を計算した。

$$\text{理論値} = \frac{\text{探索した局面数}}{\text{使用した PC の探索速度の合計}}$$

$$\text{効率} = \frac{\text{探索時間の理論値}}{\text{探索時間の実効値}} \times 100$$

5.2. 実験 1

速度差の大きな PC を 5 台使用して実験を行った。各 PC は 10base-T LAN で接続している。使用した PC の性能を表 1 に示す。各 PC の探索速度の測定は、分散処理を行わない単一 CPU 用のリバーシの人工知能のプログラムを使用した。実験は、Visual C++で作成したプログラムと、Java アプレットの 2 つでそれぞれ行った。結果を表 2、表 3 に示す。

ほとんどの状況において 5 台中最速の PC が単体で探索を行うよりも高速に探索を行うことが出来たが、分散処理の効率にばらつきが大きく現れた。これは PC の速度差が大きかったのが原因の一つだと考えられる。速度差が大きい場合、探索に時間がかかる場所に高速な PC が割り当てられれば効率が良いが、逆に低速な PC が割り当てられると無駄な探索が増え、効率が低下するためである。

52 手後の状況は効率が極端に悪いが、これは石を置ける場所が非常に少ない（平均で 5 個所程度）ために無駄な探索が多く行われたということと、探索時間が非常に短いために分散処理によるオーバーヘッドが大きく現れたことが原因である。

Visual C++で作成したプログラムと Java アプレットでは、効率は同じ傾向になったが、探索速度は C++の方が 5 倍ほど高速であった。よって、性能を重視する場合は C++を使用するのが良いといえる。

5.3. 実験 2

性能の等しい PC を用いたときの、台数に対する効率の変化を調べる実験を行った。使用した PC の性能は、CPU:Celeron 433MHz、探索速度:16505.35 局面/秒 である。なお、この実験は Visual C++で作成したプログラムでのみ行った。結果を表 4 に示す。

	CPU	探索速度（局面/秒）	
		Visual C++	Java Applet
PC1	Pentium 120MHz	2776.56	512.28
PC2	MMX Pentium 166MHz	4630.13	875.26
PC3	Pentium II 233MHz	8686.82	1723.58
PC4	Pentium II 266MHz	9678.71	1921.36
PC5	Duron 700MHz	22429.59	4296.03

表 1: 実験 1 に用いた PC の性能

状況	探索した局面数	PC5 のみ	PC1～PC5 分散処理		
		探索時間（秒）	探索時間（秒）		分散処理の効率(%)
			実効値	理論値	
4 手後	47321	2.073	1.387	0.982	70.78
16 手後	185941	7.697	4.752	3.858	81.18
28 手後	355497	15.202	8.393	7.375	87.87
40 手後	151596	6.545	6.415	3.145	49.03
52 手後	485	0.025	0.060	0.010	16.77

表 2: 実験 1 の結果（Visual C++）

状況	探索した局面数	PC5 のみ	PC1～PC5 分散処理		
		探索時間（秒）	探索時間（秒）		分散処理の効率(%)
			実効値	理論値	
4 手後	47321	10.143	7.448	5.073	68.11
16 手後	185941	37.158	24.600	19.933	81.03
28 手後	355497	78.219	44.297	38.109	86.03
40 手後	151596	33.731	31.138	16.251	52.19
52 手後	485	0.200	0.366	0.052	14.21

表 3: 実験 1 の結果（Java アプレット）

状況	探索した局面数	1 台	2 台		4 台	
		探索時間	探索時間	効率	探索時間	効率
4 手後	47321	3.109	1.772	87.70	1.306	59.49
16 手後	185941	10.946	6.374	85.86	4.526	60.05
28 手後	355497	21.110	11.997	87.98	8.602	61.35
40 手後	151596	8.692	5.598	77.63	4.602	47.22
52 手後	485	0.030	0.075	20.00	0.060	12.50

表 4: 実験 2 の結果

2 台の場合においては、52 手後の状況を除いて平均で 85%程度の効率であったが、4 台の場合では 60%程度と大きく落ちた。52 手後の状況の結果が悪いのは実験 1 と同様の原因だと思われる。

6. むすび

本論文では、ゲーム木の探索の高速化の手法として、ロビーサービス型のゲームサーバとそれに接続したクライアント PC を用いる方法を述べた。また、実験を通じ、提案システムの有効性を検証した。今後は、分散の段数を増やすノードの実装、PC の速度差が大きい場合でもある程度の効率を保つ方法などについて研究を進める。

参考文献

[1] 馬場口　登、山田　誠二　：
人工知能の基礎，昭晃堂(1999)

[2] 松原　仁　：　将棋とコンピュータ、
共立出版株式会社(1994)

[3] distributed.net : http://www.distributed.net/