6K-1

単独エージェント探索における大規模並列化手法の提案と解析

小林　義和　　岸本　章宏　　渡辺　治
東京工業大学　大学院情報理工学研究科

1　はじめに

近年，大規模な並列計算機が使えるようになり，大量の CPU と分散メモリを活用することが可能になりつつある．そのため，並列探索のアルゴリズムの研究は重要である．Hash Distributed A* (HDA*) は，A*アルゴリズムを単純に並列化するアルゴリズムであり，大量の分散メモリを効果的に活用できる．我々は，HDA*を最適マルチプルシーケンスアラインメント問題に適用し，その性能を実験的に評価した．HDA*はハッシュ関数に Zobrist 関数を用いると，利用する CPU コア数が多いとき探索オーバーヘッドが多く発生するようになり，性能が悪化する．そこで，本研究では格子座標に基づく超平面ハッシュ関数を提案し，このハッシュ関数を用いることでオーバーヘッドを低減させることができることを示す．我々の提案する手法を用いると，300 以上の CPU コアを用いても，高いスケール性を示す．

2　関連研究

HDA*は A*アルゴリズムを拡張した並列探索アルゴリズムであり，岸本らによって提唱された [1]．HDA*はハッシュ関数を用いて，各ノードの担当プロセッサを一意に決定する．新しく作られたノードは，ローカルのオープンリストには挿入されず，そのノードの担当プロセッサへと非同期に送信される．HDA*の利点は，通信がすべて非同期であり重複判定がすべてローカルで実行されるために，ローカルの仕事がなくならない限り同期オーバーヘッドが発生しないという点にある．

岸本らは HDA*をプランニングの問題に適用し，100 以上の CPU コアを用いても，速度・解答能力の双方で高い台数効果を得られることを示した．しかし，並列度をさらに大きくしたときに性能が悪化することがあることが観察された．並列度を増やすことで性能が悪化するのは，探索順序の変化が起こりやすくなるために，再探索などのオーバーヘッドが発生しやすくなることが原因と考えられる．よって，HDA*を様々な問題や環境で汎用的に利用できるようにするためには，こうしたオーバーヘッド発生の原因を理論と応用の双方から解析することが望まれる．

我々は HDA*を最適マルチプルシーケンスアラインメント問題に適用し，実験的に性能を評価する．そして，岸本らがプランニングで利用した Zobrist 関数を用いると，最適マルチプルシーケンスアラインメント問題ではスケールしないことを示す．この問題への解決策として新しいハッシュ関数を用いたノードの分配法を提案する．

3　提案方式：超平面ハッシュ関数

HDA*では各ノードのハッシュ値をプロセッサ数で法を取ることで担当プロセッサを求める．岸本らの研究では，予め初期化された乱数表を利用するハッシュ関数である Zobrist 関数 [2] を用いた．Zobrist 関数の最大の利点は，ロードバランスがよいということである．しかし実行プロセッサ数が増加すると，通信相手が多くなるために，ノードが生成されてから通信されるまでの時間が長くなる．

本研究では，本来の HDA*で用いるハッシュ関数とは異なるハッシュ関数を与える．我々が定義する超平面ハッシュ関数 Plane は，

$$\mathrm{Plane}(\boldsymbol{x}, d) := \begin{cases} \frac{1}{d}\sum x_i & \left(d \in \{1,2,3,...\}\right) \\ \frac{1}{d}\sum x_i + \left(\mathrm{Z}(\boldsymbol{x}) \bmod \frac{1}{d}\right) & \left(d \in \{\frac{1}{2}, \frac{1}{3}, ..., \frac{1}{p}\}\right) \end{cases}$$

と定義される．p は実行プロセッサ数であり，d は超平面の厚さを意味する．

超平面ハッシュ関数を用いると，あるノードから生成される子ノードが所属するプロセッサの数を全プロセッサ数よりも少なくすることができる．そのため，プロセッサ数が増えたときに通信にかかる時間が遅くなるのを防ぐことができる．

4　評価と考察

我々は提案手法の有効性を確かめるために，現実のアラインメントデータベースである BAliBASE3.0[3] より，逐次実行で 10 分以上 1 時間未満で解けた問題，及び 12 並列で 1 時間未満で解けた問題を用いて評価を行った．コスト関数には PAM250 を用いた．但し各枝のコスト値が負の値だと A*アルゴリズムでは最適解

Large-scale Parallel Search Methods for Single-agent Search.
Yoshkazu KOBAYASHI Akihiro KISHIMOTO Osamu WATANABE
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

				time (speed up)					
name	n	l	hash	$p = 12$	$p = 24$	$p = 48$	$p = 96$	$p = 192$	$p = 384$
BB12022	5	1280	Zobrist	207.30 (9.44)	105.29 (18.58)	50.47 (38.76)	27.27 (71.73)	15.90 (122.98)	> 600.0
			Plane	214.29 (9.13)	101.60 (19.25)	52.08 (37.56)	31.22 (62.65)	12.63 (154.88)	7.00 (279.55)
BBS12023	5	3481	Zobrist	308.05 (10.00)	136.58 (22.55)	70.80 (43.50)	37.46 (82.20)	82.05 (37.53)	> 600.0
			Plane	284.89 (10.81)	136.59 (22.55)	67.06 (45.92)	31.91 (96.51)	17.38 (177.22)	10.70 (287.72)
BBS11037	5	1870	Zobrist	369.53 (9.20)	164.58 (20.66)	80.80 (42.07)	41.41 (82.10)	21.49 (158.17)	> 600.0
			Plane	332.73 (10.22)	156.86 (21.67)	79.28 (42.88)	39.45 (86.17)	19.63 (173.14)	10.43 (325.97)
BBS11026	7	604	Zobrist	27.35 (11.69)	14.02 (22.79)	7.62 (41.94)	4.97 (64.27)	6.58 (48.56)	22.31 (14.33)
			Plane	27.50 (11.63)	15.02 (21.28)	7.47 (42.80)	4.21 (76.01)	2.33 (137.13)	1.77 (180.67)
BB12036	7	1499	Zobrist	170.01 (11.07)	80.90 (23.27)	41.59 (45.27)	22.13 (85.08)	22.03 (85.47)	52.16 (36.09)
			Plane	161.18 (11.68)	83.93 (22.43)	43.01 (43.77)	24.94 (75.49)	15.42 (122.04)	13.88 (135.60)
BBS12010	7	2221	Zobrist	278.49 (10.82)	138.70 (21.72)	69.97 (43.06)	79.34 (37.98)	69.77 (43.19)	115.06 (26.19)
			Plane	260.91 (11.55)	131.86 (22.85)	63.28 (47.62)	33.12 (90.97)	16.65 (180.98)	9.88 (305.05)
BBS12032	9	586	Zobrist	36.61 (11.92)	18.77 (23.25)	9.58 (45.55)	5.54 (78.80)	4.59 (95.15)	4.42 (98.65)
			Plane	37.57 (11.61)	18.72 (23.31)	9.63 (45.30)	5.19 (84.15)	3.01 (145.15)	2.25 (193.49)
BB12023	5	3593	Zobrist	542.12 (–)	372.78 (–)	138.37 (–)	68.68 (–)	34.52 (–)	> 600.0
			Plane	582.66 (–)	263.49 (–)	131.67 (–)	62.39 (–)	34.34 (–)	21.20 (–)
BB12003	8	586	Zobrist	893.53 (–)	439.81 (–)	223.75 (–)	114.16 (–)	89.70 (–)	72.78 (–)
			Plane	892.13 (–)	470.11 (–)	229.33 (–)	117.30 (–)	62.26 (–)	38.90 (–)
BBS12005	9	1815	Zobrist	588.33 (–)	299.08 (–)	147.66 (–)	89.90 (–)	57.63 (–)	39.74 (–)
			Plane	582.46 (–)	293.23 (–)	147.78 (–)	74.99 (–)	40.05 (–)	31.68 (–)
BB12032	9	613	Zobrist	1559.86 (–)	783.62 (–)	394.75 (–)	198.43 (–)	125.00 (–)	87.97 (–)
			Plane	1592.44 (–)	802.75 (–)	405.14 (–)	201.32 (–)	106.67 (–)	66.09 (–)

表 1: ハッシュ関数に Zobrist と Plane を用いた場合の比較．時間は秒．p はプロセス数，n はアラインメントの本数，l はシーケンスの長さの合計値．speed up は逐次で解けた問題のみ記述．

を求められないため，すべて正の値になるようにかさ上げをして用いる．実装には ANSI C と MPI を用い，Intel C コンパイラと MVAPICH2 でコンパイルを行った．実行には東京工業大学の TSUBAME2.0 を用いた．TSUBAME2.0 の各計算ノードは Intel Xeon X5670 で構成されており，それぞれ 54GB のメモリと 12 コアの CPU (2.93GHz, 6 Cores × 2) がある．本研究では最大 32 計算ノード (384 コア) まで利用した．

超平面ハッシュ関数を用いる場合，厚さ d を増やすと通信相手を減らせるが，同時にロードバランスを悪化させる．特にプロセッサ数が増えるとロードバランスの悪化が顕著になる．そこで，入力のシーケンスの長さの合計を l としたとき，厚さ d を

$$d' := \frac{kl}{\log p}, \quad d := \begin{cases} \mathrm{round}(d') & (d' \geq 1) \\ \frac{1}{\mathrm{round}(1/d')} & (d' < 1) \end{cases}$$

となるようにして，実験を行った．但し round は四捨五入であり，k は定数で d 今回は $k = 0.003$ とした．

表 1 はハッシュ関数によるスケーラビリティの違いを示す．Zobrist 関数を用いた場合，96 並列以降では台数効果が得られなくなるが，超平面ハッシュ関数を用いると 384 並列でも高い台数効果を示している．

これは探索オーバーヘッドが原因である．Zobrist 関数を用いた場合，並列度が大きくなるとノードが生成されてから通信されるまでに時間がかかるようになるため，本来の探索順序とは大きく異なる順序で探索が行われ，非最適なノード展開が起こりやすくなり，探索オーバーヘッドを大きく増加させる．それに対し，超平面ハッシュ関数を用いた場合ではノードが通信されるまでの時間が Zobrist 関数を用いた時より短いため，オーバーヘッドの増加が小さい．

5 まとめ

我々は大規模並列探索アルゴリズムである HDA*を最適マルチプルシーケンスアラインメント問題に適用し，ハッシュ関数として Zobrist 関数を用いると探索オーバーヘッドが発生してスケール性が悪くなることを示した．そして，その問題を解決するために超平面ハッシュ関数を提案し，これを用いるとより高いスケール性を示すことを示した．

今後の課題として，より多くの問題に適用可能なハッシュ関数の設計が挙げられる．超平面ハッシュ関数はマルチプルアラインメントの特徴を利用したハッシュ関数である．問題に非依存で，通信相手を減らすハッシュ関数が設計出来れば，HDA*は多くの問題で高いスケール性を示すと考えられる．

参考文献

[1] A. Kishimoto, A. Fukunaga, and A. Botea. Scalable, parallel best-first search for optimal sequential planning. In *Proc. ICAPS*, pp. 10–17, 2009.

[2] A.L. Zobrist. A new hashing method with application for game playing. Technical Report 88, University of Wisconsin, 1970.

[3] J.D. Thompson, P. Koehl, R. Ripp, and O. Poch. BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function, and Bioinformatics*, Vol. 61, No. 1, pp. 127–136, 2005.