2E1-1

大規模並列探索を利用した古典的プランニング・システムについて

Massively Parallel Search and Its Application to Classical Planning

岸本 章宏[*1*2] Alex Fukunaga[*3] Adi Botea[*4]
Akihiro Kishimoto

[*1]東京工業大学大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

[*2]科学技術振興機構 さきがけ
JST PRESTO

[*3]東京大学大学院総合文化研究科
Graduate School of Arts and Sciences, University of Tokyo

[*4]NICTA and The Australian National University

Hash Distributed A* (HDA*) is an efficient parallel A* algorithm that asynchronously distributes work among the nodes by a global hash function. This paper evaluates the performance of HDA* with up to 1024 cores on a distributed memory cluster. Our experimental results show that HDA* scales fairly well in domain-independent classical planning.

1. はじめに

プランニング・システム (プランナー) は，PDDL や STRIPS などのプランニング言語で記述された問題と目標に対して，目標を達成する解 (プラン) を自動的に返すシステムである．プランニングの応用分野として，宇宙船や自律ロボットなどの制御，工場での製造工程の解析，ゲームなど多くのものが挙げられる [Ghallab 04] ので，プランニングは人工知能における代表的な課題として盛んに研究されている．

古典的プランニングは，プランニングの中で最も単純であるが，それでも計算量の観点から難しい課題である [Bylander 94]．このため，探索アルゴリズムと自動抽出したヒューリステックを利用して，最適なプランを求めるドメイン非依存型のプランナーの研究は，文献 [Helmert 07, Helmert 09] などを代表として盛んに行われているのにもかかわらず，古典的プランニングの難しい問題の多くは，最新技術を用いても解けないのが現状である．

プランナーの速度および解答能力を向上させる方法の一つに，アルゴリズムの並列化がある．特に，分散メモリ環境では，クラスター計算機が最近は普及しているだけでなく，グリッドやクラウド環境の研究も盛んである．これらの環境を利用すれば，プランナーは大多数の CPU コアだけでなく，大容量のメモリも利用できる．

我々が以前に開発した HDA*(Hash Distributed A*) アルゴリズム [Kishimoto 09] では，最適解を保証するプランナーが通常利用している探索手法である A* アルゴリズム [Hart 68] を並列化した．古典的プランニングの実験では，HDA* は 8CPU コアの共有メモリマシンや合計 128 コアの CPU を持つ分散メモリ環境で有効であった．

本論文では，[Kishimoto 09] にある HDA* を 1024CPU コアを持つ分散メモリ環境上に実装し，古典的プランニングの問題を実際にプランナーに解かせることで，性能を評価する．

本論文の構成は以下の通りである．第 2 節では，HDA* アルゴリズムについて概説する．第 3 節では，実験結果について議論し，第 4 節でまとめを行う．

連絡先: 東京工業大学大学院情報理工学研究科 数理・計算科学専攻〒 152-8552 東京都目黒区大岡山 2-12-1-W8-25
Email:kishimoto@is.titech.ac.jp

2. HDA* アルゴリズム

最適解を保証するプランナーは，A* アルゴリズム [Hart 68] を利用している．古典的プランニングにおいて最も計算時間を要するところは，A* 探索を行い，プランを達成する目標状態を発見する部分であるので，A* を並列化すれば，プランナーは高速になる．また，A* は以前に展開した状態をすべて保持するので，状態数の組み合わせ爆発が生じるプランニングのような探索空間では，メモリをすぐに使い切ってしまう．分散メモリ環境で並列化すれば，逐次アルゴリズムよりも利用できる総メモリ量が増えるので，プランナーの解答能力は改善する．

A* は，ある状態を以前に展開したことのあるかどうかの判定にクローズドリストを利用し，新しい状態の展開にオープンリストを用いる最良優先探索である．一方，HDA*[Kishimoto 09] では，各プロセッサはプロセッサ間で互いに素な形でクローズドリストとオープンリストの一部を管理し，全体で一つの大きなクローズドリスト (またはオープンリスト) とみなせるデータ構造を保持する．このため，分散メモリ環境の利用により増加したメモリを，プロセッサ間で重なりのないクローズドリストとオープンリストに割り当てられるので，HDA* は，メモリのスケーラビリティが良い．

HDA* では，各プロセッサ P_i が並列に状態の展開を行う．P_i が新たに作り出した状態 S は，S をクローズドリスト (またはオープンリスト) に保持すべきプロセッサ P_j に必ず送信する．P_j の決定にはハッシュ関数を利用する．このようにすれば，S が以前に展開済みの状態であれば，P_j で必ず同一状態の判定が行える．また，P_j に状態を送ってしまえば，P_i は別状態の展開が行えるので，プロセッサ間で同期を取る必要がない．

3. 実験結果

HDA* を東京工業大学の TSUBAME 上に実装し，性能評価を行った．TSUBAME の各ノードは，Sun Fire X4600 で

構成されており，各ノードには 32GB のメモリと 16 コアの CPU(デュアルコア Opteron 2.4 GHz × 8) がある．本論文の実験では，最大 64 ノード (1024 コア，合計メモリ 2TB) まで利用した．対象プログラムには，高性能な古典的プランナーの一つである [Helmert 07] の手法に基づく Fast Downward プランナを利用し，Fast Downward 上に MPI を用いて HDA* を実装した．性能評価実験では，国際会議 ICAPS で隔年に行われているプランニング大会 (IPC) の過去問 (第 3 回，第 4 回，および第 6 回) のいくつかを TSUBAME 上で HDA* に解かせ，実行時間を計測した．

逐次 A* で解けた問題に対して，HDA* は，16 コアでは 9-12 倍，64 コアでは 33-65 倍，128 コアでは 64-98 倍，512 コアでは 165-371 倍，1024 コアでは 55-651 倍の高速化率を達成した．HDA* は，512 コアまでは，CPU コアを利用すればするほど，高速化を達成している．しかし，1024 コアを利用した場合には，いくつかの問題では，512 コアよりも遅くなった．1024 コアを利用した場合には，探索オーバーヘッド (逐次探索に比べて，並列アルゴリムが余計な状態を展開してしまう割合) の急激な増加が見受けられた．最悪の場合の探索オーバーヘッドは，512 コアでは約 150%程度であるのに対し，1024 コアでは約 530%であった．

探索オーバーヘッド増加の原因の可能性の一つとして，1 コアでは解くのが難しかった問題が，512 コアでは非常に簡単になってしまい，すでに並列効果が出にくい状態であることが考えられる．実際に，A* をベースとするアプリケーションを並列化する際には，n コア利用した場合に逐次 A* に比べて何倍速くなったかを調べる高速化率だけでは，並列アルゴリズムの性能評価の基準として不十分である．逐次 A* は，並列アルゴリズムよりも利用できるメモリが少ないため，難しい問題に対しては解を求められず，ノード内のメモリをすべて使い果たして終了してしまう[*1]．

ある問題 P を n コアを用いて解くのにかかった時間を t_n とし，P を解くのに最低限必要な CPU コア数を用いた場合の実行時間を t_{min} とする．P に対して，n コアの相対効率を $\frac{t_n}{t_{min}}$ を定義する．このように定義すれば，逐次 A* が問題を解けない場合でも HDA* が並列化による効率がどの程度得られているのかが調べられる．

図 1 に，1024 コアまでの HDA* の相対効率を示す．この図では，1 コアで解ける問題のいくつかに対しては，1024 コアでは 512 コアよりも相対効率が上昇しているが，解くのに少なくとも 128 コア (総メモリ 256GB) が必要な問題では，Sokoban-26 を除いて，1024 コアを利用しても引き続き高速化を達成していることが分かる．例えば，求解のためには少なくとも 512 コア必要であった問題に対しては，1024 コアの相対効率は 0.5-0.7 であるので，512 コアに対して 1.4 倍から 2 倍程度の高速化を達成している．

HDA* では，並列化による総メモリ量の増加により，難しいプランニング問題に対する解答能力も向上した．例えば，第 6 回 IPC にある倉庫番問題は 30 題から成るが，逐次 A*(メモリ 32GB) では 22 題しか解けなかったのに対し，512 コア (メモリ 1TB) での解答数は 28 題に増加した．

1 本論文の実験では，逐次 A は 128GB のメモリを持つ Opteron 2.6GHz のマシン上で実行し，2.4GHz の Opteron における推測実行時間に変換している．128GB のメモリを持つ環境で逐次 A* では解けないが，より多くのメモリを利用できる HDA* では解ける問題は，数多く存在した．

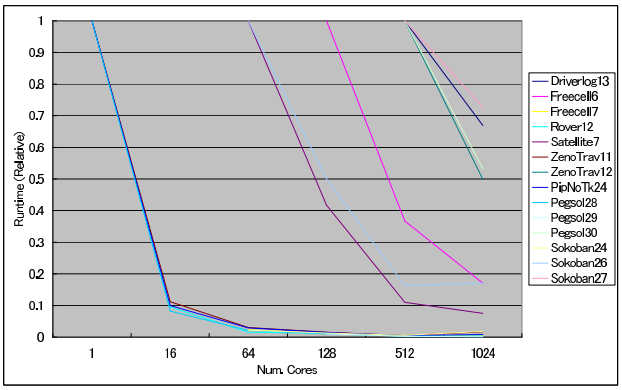

図 1: HDA* の相対的な効率

4. まとめ

本論文では，並列アルゴリズム HDA*[Kishimoto 09] を利用したプランナーを実装し，1024 という大規模な CPU コア数で性能評価を行った．その結果，512 コアまでは高い高速化率を得られただけでなく，メモリを大量に利用することでプランナーの解答能力も向上した．1024 コアでは，逐次プランナーでも解けるような簡単な問題に対しては，探索オーバーヘッド増加による性能低下が見受けられたが，並列化によって初めて解けた問題に対しては，引き続き高い高速化率を達成した．

参考文献

[Bylander 94] Bylander, T.: The Computational Complexity of Propositional STRIPS Planning, *Artificial Intelligence*, Vol. 69, No. 1–2, pp. 165–204 (1994)

[Ghallab 04] Ghallab, M., Nau, D., and Traverso, P.: *Automated Planning: Theory and Practice*, Morgan Kaufmann (2004)

[Hart 68] Hart, P. E., Nilsson, N. J., and Raphael, B.: A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics*, Vol. 4, No. 2, pp. 100–107 (1968)

[Helmert 07] Helmert, M., Haslum, P., and Hoffmann, J.: Flexible Abstraction Heuristics for Optimal Sequential Planning, in *17th International Conference on Automated Planning and Scheduling*, pp. 176–183 (2007)

[Helmert 09] Helmert, M. and Domshlak, C.: Landmarks, Critical Paths and Abstractions: What's the Difference Anyway?, in *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS 2009)*, pp. 162–169 (2009)

[Kishimoto 09] Kishimoto, A., Fukunaga, A., and Botea, A.: Scalable, Parallel Best-First Search for Optimal Sequential Planning, in *19th International Conference on Automated Planning and Scheduling (ICAPS-2009)*, pp. 201–208 (2009)