7F−3

TMSの確率的アルゴリズムによる並列インプリメンテーション

岩山登 佐藤健
(財)新世代コンピュータ技術開発機構

1 はじめに

Doyle の Truth Maintenance System [1](以下, TMS) は, 信念間の制約を justification と呼ばれるルールで表し, justification に矛盾しない信念の状態を導く処理系である. TMS は仮説推論や非単調推論などに用いられ, 多くの研究がなされている. 我々の目的は, TMS を疎結合並列計算機で実現し短時間で計算することにある. 疎結合並列計算機で実現するには通信が少なくなければいけないが, 過去に提案されたアルゴリズム [3] [4] [5] はアーキテクチャとして密結合並列計算機を暗に仮定しており, 通信量が多く疎結合並列計算機での実現には向かない.

そこで通信が少ないことに重点を置き, 計算の途中ではまったく通信する必要のない実現方法を提案する. それは, 非決定的アルゴリズムを確率的アルゴリズムを用いて並列化したものである.

2 確率的アルゴリズムとその並列化

ここでいう確率的アルゴリズムとは, 非決定的アルゴリズムの探索の途中で取り得る計算のパスが複数あるとき, そのうちの一つをランダムに選ぶアルゴリズムである. もし選んだ計算のパスが失敗であることが分かったら, 初期状態からもう一度ランダムに選び直し, 1つ解が見つかるまでこれを繰り返す. 確率的アルゴリズムの並列化は, 各プロセッサで確率的アルゴリズムを用いて同じ問題を計算すればよい. 1つ解を見つければよいので, あるプロセッサで解が見つかったらそのことを他のプロセッサに知らせ計算を停止させる. 各プロセッサで異なる乱数発生器を用いれば, 確率的にはすべての探索パスを調べることができる. これによって並列確率的アルゴリズムは OR 並列探索を行っているとみることができる.

並列確率アルゴリズムは, 初期条件を各プロセッサにコピーした後は, 1つ目の解が見つかるまでまったく通信する必要がない. しかし, 同じ探索パスを重複して選んでしまうかもしれないという欠点がある. 確率的アルゴリズムの利害得失をみるために, 確率的アルゴリズムの計算量について考える. 確率的アルゴリズムの計算量を議論するとき, 同じ入力に対しても計算する度にその時間は異なる可能性があるので, 期待値としての計算時間を考えなくてはいけない. 以下では, いくつかの仮定をおいて確率的アルゴリズムの期待計算時間を計算してみる.

1つの探索パスに解が存在する確率を p とする. 1プロセッサで1解を得るまでの試行回数 (解を得るまでに失敗した回数 +1) の期待値は $1/p$ であり, 1試行に要する時間 (t) がパスによらず一定であるとすれば, 1解を求めるのに要する期待時間量 T_1 は t/p である. n プロセッサを用いたとき少なくとも1つのプロセッサで解が得られる確率 $p_n = 1-(1-p)^n$ であるから, n プロセッサで1解を得るまでの試行回数の期待値は

$$\frac{1}{1-(1-p)^n}$$

で, 1解を求めるのに要する期待計算時間量 T_n は

$$\frac{t}{1-(1-p)^n}$$

となる. すると台数効果は,

$$\frac{T_1}{T_n} = \frac{1-(1-p)^n}{p} \tag{1}$$

となる. よって

$$\lim_{p\to 0}\frac{T_1}{T_n} = n.$$

これは, 問題が非常に難しく探索空間が大きい場合, n 台のプロセッサで重複なく探索できることを表している. また,

$$\lim_{p\to 1}\frac{T_1}{T_n} = 1$$

で, 問題が簡単で解がたくさん存在する場合はすぐに解が見つかってしまうので台数効果は上がらない.

さらに

$$\frac{T_n}{T_{2n}} = \frac{1-(1-p)^{2n}}{1-(1-p)^n} = 1+(1-p)^n$$

でプロセッサ数を2倍にしてもスピードアップは $1+(1-p)^n$ 倍にしかならず, プロセッサを増やすと選択の重複が多くなることを示している.

3 TMS の非決定的アルゴリズム

TMS は, node 集合, justification 集合, nogood 集合からなる. node はエージェントの信念を, justification は信念の間の関係を, nogood は信念の組み合わせの禁止を表す. 例えば, justification $\langle n, \{i_1, i_2\}, \{o_1, o_2\}\rangle$ はもし i_1 と i_2 を信じていて o_1 と o_2 をいずれも信じていないならば, n を信じなければならないことを表し, nogood $\{i_1, i_2\}$ はもし i_1 と i_2 を信じているならば, 矛盾であることを表す.

以前我々は, 与えられた TMS に対し admissible state(信念の状態) を一つ計算する非決定的アルゴリズムを提案し, その正当性の証明と並列論理型言語 KL1 を用いたインプリメントを行った [2]. そのアルゴリズムの要点は, 一つの justification が成り立つことを仮定し, その仮定によって派生的に得られた情報を元にボトムアップに admissible state を作り上げていくことである.

A Parallel Inplementation of TMS by Randomized Algorithm
Noboru IWAYAMA, Ken SATOH
Institute for New Generation Computer Technology

アルゴリズムの非決定性は, justification を仮定するときに仮定できる justification が複数あることから生じる.

Doyle の定義では, 現在の信念の状態に矛盾する justification が付け加わったら矛盾を解消するために信念の状態を変化させ, 新しい justification の集合全体に矛盾しない信念の状態を求めている (Dependency Directed Backtracking). しかし, ここでは justification はすべて最初から与えられているものとしている. TMS にこのような制約を課したとしても, (Dynamic) Constraint Satisfaction Problem [6], Abduction [7] などの応用がある.

4 TMS への確率的アルゴリズムの適用 (実験と考察)

TMS の非決定的アルゴリズムを確率的アルゴリズムを用いて実現した. 実現は ICOT で開発した並列推論マシン マルチ PSI/V2 上 [8] で行った. TMS でクィーン問題を記述し実験を行った結果, 台数効果で理論値と実験値がおおよそ一致することを確認した (図 1). 理論値は 2節の (1) 式に基づいている. 実験には例題として 11 クィーン問題を用い, 期待値を調べるため 100 回求解を繰り返した. また, p を理論的に求めるのは容易でないので, 理論値の計算には実験的に求めた値 ($p = 1/15.3$) を用いた. プロセッサ 1 台の場合, 100 回の求解に約 3800 秒かかっている.

実験の結果が 2節の計算と一致したのは, 1 試行に要する時間は我々のインプリメンテーションの場合選ばれたパスによらず一定であるからだと考えられる. 我々のインプリメンテーションでは justfication 集合と nogood 集合に現れる node すべてを KL1 プロセスとして表現し, チェックする必要のなくなった node を表すプロセスが delete されることによって計算が進む. 矛盾が生じた場合, その時点で残っているプロセスを delete して新しくすべてのプロセスを作り直す. それぞれのプロセスはたかだか定数回しかアクティブにならないので, 1 試行に要する reduction 数は justification 集合と nogood 集合に現れるのべ node 数に比例し, 確率的な計算パスの選択に関わらず一定になる.

5 おわりに

TMS の確率的アルゴリズムによる並列化について述べた. 確率的アルゴリズムによる並列化を, Prolog のバックトラック, ある種の組み合わせ問題について適用した研究があり [9] [10], 問題によっては linear 以上の台数効果がでることもある [10].

謝辞

研究の機会を与えて下さった ICOT 渕所長, 古川次長, 長谷川部長代理, 相場第 4 研究室長代理, また, 杉野栄二氏をはじめとする討論ご助言頂いた ICOT の方々に感謝する.

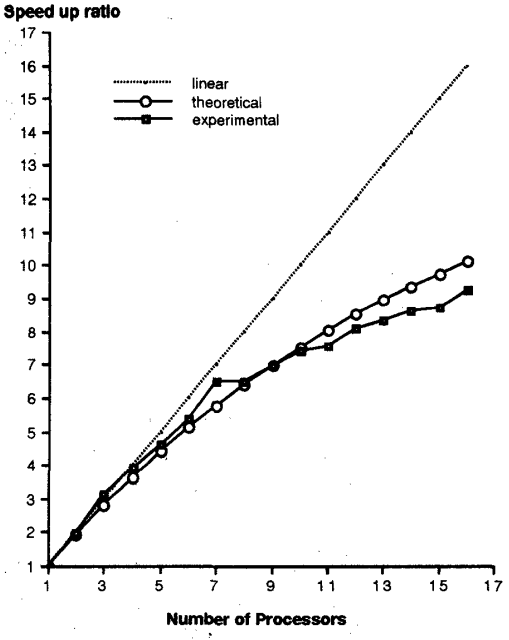

図 1: 11queen 問題の台数効果

参考文献

[1] Doyle, J.: A Truth Maintenance System, *Artificial Intelligence*, **12**, pp. 231 – 272 (1979).

[2] Satoh, K., Iwayama, N., Sugino, E., Konolige, K.: An Inplementation of TMS in Concurrent Logic Programming Language: Preliminaly Report, Tech. Rept. 568, ICOT, (1990).

[3] Petrie, C. J. Jr.: A Diffusing Computation for Truth Maintenance, *Proc. ICPP-86*, pp. 691 – 695 (1986).

[4] 久野禎子, 久野靖: 並列オブジェクト指向言語を用いた TMS の再構成, 人工知能学会誌, 4, pp. 62 – 69 (1989).

[5] Fulcomer, R. M., Ball W. E.: Correct Parallel Status Assignment for the Reason Maintenance System, *Proc. IJCAI-89*, pp 30 – 35 (1989).

[6] Mittal, S., Falkenhainer, B.: Dynamic Constraint Satisfaction Problems, *Proc. AAAI-90*, pp 25 – 32 (1990).

[7] Kakas, A. C., Mancella, P.: On the relation between Truth Maintance and Abduction, *Proc. 3rd Nonmonotonic Reasoning Workshop*, pp. 158 – 176 (1990).

[8] Nakajima, N., Inamura, Y., Ichiyoshi, N, Rokusawa, K., Chikayama, T.: Distrubuted Implementation of KL1 on the Multi-PSI/V2, *Proc. ICLP-89*, pp. 436 – 451, (1989).

[9] Janakiram, V. K., Argawal, D. F., Mehrotra, R.: Randominzed Parallel Algorithm for Prolog Programs and Backtracking Applications, *Proc. ICPP-87*, pp. 278 – 281 (1987).

[10] Mehrotra, R., Geringer, E. F.: Superliner Speedup Through Randomized Algorithms, *Proc. ICPP-85*, pp. 291 – 300 (1985).