Boolean Buchberger Algorithm とその並列化 (4) - 実現と評価

毛受 哲, 岩山 登, 佐藤 健, 佐藤 洋祐
(財) 新世代コンピュータ技術開発機構

1 はじめに

ブール等式の集合が与えられたとき, Buchberger Algorithm を改良したアルゴリズムによって Boolean Gröbner base を求めることができる [1]. 我々は, このアルゴリズムの並列版を Multi-PSI 上で実現することを目指し, 逐次版アルゴリズムの解析を行い [2], それをもとに並列モデルを作成し [3], 並列論理型言語 KL1 で記述実験を行っている. ここでは並列版の Multi-PSI 上における実現と, いくつかの例に対して行った実験による解析について報告する.

2 並列アルゴリズム

並列アルゴリズムは次のような部分から成っている [3].

1. 入力制約を内部表現に書き換え, 最大単項がルールで書き換えられなくなるまで書き換えたものを, ルール候補式とする.
2. ルール候補式の中で最も最大単項の小さい式を新しいルールとし, そのルールで自分以外のすべてのルールの書き換えを行う. その書き換えによって最大単項が書き換わった古いルールはルール集合から取り除き, ルール候補式とする.
3. 新しいルールと残った古いルールから, クリティカル・ペアとセルフ・クリティカル・ペアを作りルール候補式とする.
4. すべてのルール候補式に対し, それぞれの最大単項がルールによって書き換えられなくなるまで書き換える.

並列アルゴリズムは, 入力制約があれば 1 を行い, なければルール候補式がなくなるまで 2 から 4 を繰り返す.

3 KL1 による実現

Multi-PSI 上で効率の良い実現をすることが目的なので, なるべくプロセス間通信を少なくするような実現を行った. [3] のモデルにより, ルール候補式の最大単項を書き換えできなくなるまで書き換える部分 (前節並列アルゴリズムの 4) を並列化するので, 各プロセスにはルール候補式とルールが必要である. ルールは毎回送ると通信量が多くなるので差分を送ることにし, 最初の実現を行った. その後, 以下の改良を行った.

改良 1 前節の並列アルゴリズムの 2 において取り除かれたルールから作られたルール候補式は, 分散する前に取り除くようにした. そのような式を分散してから取り除くと仕事は多少分散されるが, 通信量が増えるし仕事が均等に分散されなくなる恐れがある.

改良 2 $x = A$ のように左辺が変数 1 つであるようなルールは, 他の式のその変数に右辺を代入するだけであり, クリティカル・ペアを作ることもない. そこでこのような形のブール等式がルールになったとき, 持っているすべてのルールとルール候補式に代入を行い, そのルールは別に持つことにする. そしてブール制約式が新しく入力されると最初に代入を行うようにする. このようにすると, 各プロセスがそのルールを持つ必要もないし, クリティカル・ペアを作ろうとしなくてすむ.

改良 3 ルールの右辺が 0 の場合, セルフ・クリティカル・ペアは作れない. また, 右辺が 0 のルール同士からはクリティカル・ペアは作れない. そこで右辺が 0 であることのチェックをするようにする. これは逐次版にも有効である.

4 評価

6 クイーン問題, 5 入力 3 出力のカウンター回路の問題, グラフの辺を色塗りする問題を解いた結果について考察する. 6 クイーン問題に関しては, 改良前の版に対する実験結果についても述べることにする.

4.1 6 クイーン問題

盤の各マスをそれぞれ変数で表し, クイーンを置くなら 1, 置かないなら 0 を割り当てることにし, 縦・横・斜めのすべての制約を単純に記述することにする. この問題は, 0, 1 を各変数に割り当てていく高速な方法がいろいろ研究されている. Boolean Buchberger

---

Boolean Buchberger Algorithm and its Parallelization (4) - Implementation and Evaluation
Satoshi Menju, Noboru Iwayama, Ken Satoh, Yosuke Sato
Institute for New Generation Computer Technology

Algorithm はこの問題の全解を表す Gröbner base を求めるため非常に時間がかかるが，評価に使う問題としては適当な難しさである．変数の数は 36，制約式の数は 296 である．次に改良前，改良後の実行時間を示す．単位は秒である．

	1台	2台	4台	8台	12台
改良前	5421	3481	2534	2216	1821
改良後	3665	2506	1777	1654	1606

プログラムは逐次部分と並列部分からなるので，単純にそれぞれの仕事量を a, b とすると，n 台プロセッサで実行したときの実行時間は $a + b/n$ で近似することができる．定数を計算してみると，改良前のプログラムでは $a = 1600, b = 3810$，改良後では $a = 1340, b = 2310$ が求まる．改良前のプログラムは逐次プログラムを並列モデルに合わせて並列化しているので，実行時間の近似式から「並列部分が 70%」という逐次プログラムの解析により求まった値に近い数字が得られる．少し小さいのは，分散のためのオーバーヘッドや，ルール候補式が 0 かどうかのチェックを分散前に行っているため，そして必ずしも均等に仕事が分散されていないためであろう．一方，改良後のプログラムでは逐次に行う仕事が増えているので，実行時間は短くなっているが「並列部分が 63%」という値になっている．

4.2 グラフの彩色問題

三角形と五角形に対し，三角形の各頂点と五角形の各頂点を辺で結んだグラフ (図 1) を考える．このグラフの辺を 2 色で色塗りすると，必ず単色で塗られた三角形が存在することを証明する [4]．ブール制約で表現する場合は，辺を変数で表し，0 と 1 を 2 色に対応させる．そして単色三角形がないような制約を記述して解を求めるのに失敗すれば，必ず単色三角形が存在することを証明できたことになる．この問題の変数の数は 23，制約式の数は 62 である．

1台	2台	4台	8台	12台
696.8	618.1	511.1	440.0	426.8

この程度の大きさの問題になると，分散のオーバーヘッドをカバーできなくなるので台数効果が小さくなることがわかる．

4.3 カウンター回路の問題

それでは，さらに小さい問題ではどうであろうか．5 つの入力の 1 の数を 3 桁の 2 進表示にして出力する回路を，31 変数，26 式の問題として記述し，入力も出力も変数のままにして一般解を求めることにする．

1台	2台	4台	8台	12台
16.65	18.63	16.89	16.11	16.08

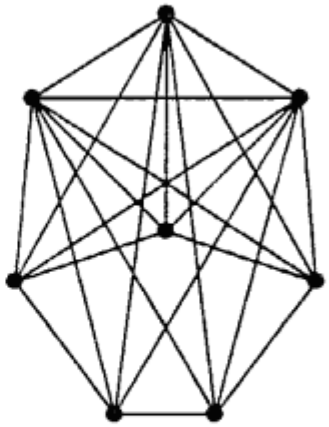

図 1: 彩色問題のグラフ

このぐらい小さい問題になると，2 台や 4 台では通信や分散のオーバーヘッドの方が大きく，1 台より遅くなっている．その後，少しずつ速くなってはいるが，ほとんど効果は出ていない．

5 まとめ

Boolean Buchberger Algorithm の並列化モデル [3] を，KL1 言語により Multi-PSI 上に実現した．最初の実現では逐次版の単純な並列化であったため，3 倍近い台数効果が得られた．4 倍までいかなかったのは，通信にかかる時間や余分な仕事が増えるのと，必ずしも均等に仕事を分散することができないためと考えられる．その後，分散しなくていい仕事を省く等の改良を行った結果，処理時間は短くなったが台数効果は 2 倍強になった．また，小さい問題に対してはオーバーヘッドをカバーするほどの分散ができないため，あまり台数効果が出ないという予想も実験により確認した．

今後の課題として，他の並列化モデルの検討，ブール多項式に対する他のアルゴリズムの並列化の検討が挙げられる．

参考文献

[1] 佐藤洋祐 他，Boolean Buchberger Algorithm とその並列化 - (1) 基礎理論，情報処理学会第 43 回全国大会論文集 7N-3 (1991).

[2] 岩山 登 他，Boolean Buchberger Algorithm とその並列化 - (2) 動作解析，情報処理学会第 43 回全国大会論文集 7N-4 (1991).

[3] 佐藤 健 他，Boolean Buchberger Algorithm とその並列化 - (3) 並列アルゴリズム，情報処理学会第 43 回全国大会論文集 7N-5 (1991).

[4] G. Pólya et al., Notes on Introductory Combinatorics, Birkhauser Boston, Inc. (1983).