PVMによるSATアルゴリズムの高速化

4F-03

梅本 潤　　宮崎修一　　岡部寿男　　岩間一雄
京都大学 情報学研究科

1 はじめに

本研究では, ネットワークを利用した並列化により, 局所探索法での充足可能性問題(SAT)の高速化を目指す. SATは基本的な組み合わせ問題であり, 様々な問題をSATに変換して解くことも多く, 非常に重要度の高い問題である. 従って, 高速SATアルゴリズムの開発は重要な研究課題となっており, 現在最も有効な手法の一つとして局所探索法 [2, 3, 5] が挙げられる.

現在は計算機の性能向上により計算パワーが増大しているが, 余った計算パワーの有効利用が大きな問題となっている. また, インターネットが普及し通信速度は向上し, 信頼性も増している. このような状況のなかで, ネットワークを利用した分散計算は非常に効果的である. そこで本研究ではネットワークに接続された異機種UNIXコンピュータ群を, 単一の並列コンピュータとして利用することを可能にするソフトウェアシステムであるPVM(**P**arallel **V**irtual **M**achine)[1]を利用する. これによって, 多数のコンピュータの持つ計算パワーを, 一つの大規模計算問題に結集して処理を行うことが出来る. PVM3は無料で配布されており, 先端科学分野における大規模計算のためのソフトウェアとして世界中で利用されている.

本研究において, 京都大学内で最大70台ほどの規模で実験を行った. また, 広島大学, 九州大学の協力により大学間を結んでの実験も行った.

本稿ではPVMでの並列化による高速化に関する計算機実験と, プログラムの性能を客観的に評価するため, *2nd DIMACS Implementation Challenge*[4] のベンチマークを行い, 実用性を示した.

2 SATおよび局所探索法

SATとは和積形論理式(CNF式)fが与えられたとき, fの全ての項を1にする割当が存在するかどうかを問う決定問題である. SATに対する高速アルゴリズムとして90年代の初めに開発された局所探索法 [2, 3, 5] がある. 局所探索法は現在の変数割当から, より多くの項が充足されるように, ある変数の割当を変える(フリップする)ことを繰り返して充足解に到達しようとするものである.

どの変数をフリップしてもより良い割当が存在しない変数割当を局所解といい, 局所解からの脱出法にいくつかの手法がある. 本研究では, フリップの際に良さが同等, あるいは悪い変数割当へのフリップも許したGSAT[2, 3] を使用する. GSATにはMAXFLIPS, MAXTRIESという2つのパラメータが必要である. ランダムに初期割当を選び, フリップをMAXFLIPS回行うことを1TRYという. GSATはTRYをMAXTRIES回行う. この過程で充足解が見つかれは, その時点で実行は終了となる. GSATのプログラムは [6] で使われたものを使用する.

A parallel local search algorithm for SAT using PVM
Jun Umemoto, Shuichi Miyazaki, Yasuo Okabe, and Kazuo Iwama
School of Imformatics, Kyoto University

3 GSATの並列化

GSATの各TRYは互いに独立である. 従って, 複数のTRYを並列化することが可能であり, 並列化されたGSAT間には依存関係がなく, 通信は解けたかどうか程度で済むため非常に並列処理に向いていると考えられる. この様なGSATの性質から, 本研究ではGSATプログラムがPVMによる並列化で高速化できると考え, 実験によりそれを示す.

具体的な実装は, マスターとスレーブの2つのプログラムからなる. スレーブプログラムはGSATそのものであり, マスタープログラムがGSATをまとめる役割を果たす. 各GSATは1 TRY毎に成功したかどうかをマスターに通信する. マスターは通信を受け, 成功なら各GSATを終了させ実行終了となり, 失敗の通信ならTRYの回数を数えて設定のMAXTRIES回を越えた時点で実行終了し, 解は得られなかったということになる.

通信のコストを考えなければ, 並列度(マシン台数, 性能は同じとする)がnとすると, 並列GSATは通常のGSATに比べて同じ時間でn倍のTRYを実行できる. 同じ回数TRYを実行すれば成功率は同じなので, 並列GSATは通常のGSATに比べ, 同じ成功率を得ようとする場合には並列度nに比例して高速化されることになる.

4 計算機実験

4.1 1秒あたりのフリップ回数に関する実験

前節で述べたように, 通信のコストを除けば並列GSATはGSATに対し, 並列度に比例して高速化できると考えられる. そこで計算機実験により通信のコストも含めて実際にどの程度高速化されるか確かめる.

また, LAN内での通信とより大きな範囲での通信ではコストがかなり違うと考えられるので, LAN内での通信のみの場合とLAN外との通信も含む場合で1秒あたりのフリップ回数を比較する.

実験では100変数と300変数のランダム例題100個ずつに対して, MAXFLIPS = (変数数) × 10, MAXTRIES = (ホスト数)× 10 の並列GSATを実行した.

図1は並列度に対する1秒あたりの平均フリップ回数, 図2はLAN内での通信のみの場合とLAN外との通信も含む場合で1秒あたりのフリップ回数の比較である.

LAN内の実験では本研究グループのマシンのみを使用した. LAN外では, GSATを実行するマシンはLAN内の場合と同じで, マスタープログラムだけを広島大で実行した. つまり, 通信は全て京都大学と広島大学の間で行われている.

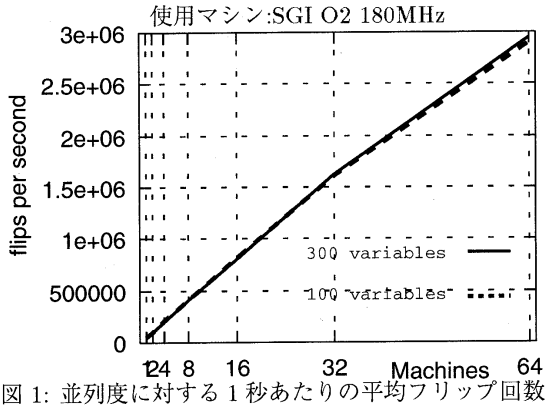

図 1: 並列度に対する 1 秒あたりの平均フリップ回数

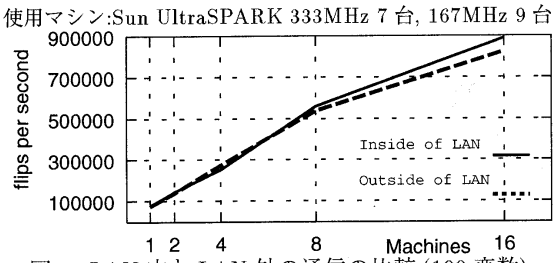

図 2: LAN 内と LAN 外の通信の比較 (100 変数)

図1の実験では, 100 変数, 300 変数ともにほぼ並列度に比例して高速化されている. ただし, 並列度が 64 まで上がると速度向上率が低下しているので, 並列度が上がって通信の頻度が増えた結果と思われる.

図2の実験では, LAN 内を通る通信も LAN 外 (京都大学, 広島大学間) を通る通信でもほぼ同様の高速化が得られている. この実験で使用マシン 16 台での速度の低下は, マシンの性能にばらつきがあるためである.

2つの実験から, GSAT の並列化では通信の影響は大きくなく, ほぼ予想通りに高速化できると考えられる. 特に, 1つ1つの通信のサイズが小さいため遅い通信路でも大きな影響はみられない. 通信の頻度が高い場合は多少影響を受けるが, SAT の問題のサイズが大きいほど通信の頻度は下がるため大きな問題ではないと言える.

4.2 DIMACS ベンチマーク

次に, 並列 GSAT の速度を客観的に評価するため, *2nd DIMACS Implementation Challenge*[4] の SAT ベンチマークに対する実験を行った. 並列 GSAT の並列度は 64 である. [4] に紹介された 7 つの論文の中で, アルゴリズムの近い Selman らの SASAT との比較を表 1 に示す. 表中の値は実行時間 (秒) であり, × は解けなかったことを表す. SASAT は GSAT に random walk という成功率向上の工夫がされたものであり, GSAT より優れていると言える.

表 1: DIMACS ベンチマークに対する結果 (単位:秒)

例題	変数	項数	並列化	SASAT
aim-100-2_0-yes1-1	100	200	5	×
aim-100-2_0-yes1-2	100	200	9	13,929
aim-100-2_0-yes1-3	100	200	1	22,500
aim-100-2_0-yes1-4	100	200	3	×
f400.cnf	400	1,700	7	60
f800.cnf	800	3,400	×	27,000
g125.17.cnf	2,125	66,272	2	453,780
g125.18.cnf	2,250	70,163	2	480
g250.15.cnf	3,750	233,965	4	120
g250.29.cnf	7,250	454,622	10	398,820
ii32b3.cnf	348	5,734	1	5,400
ii32c3.cnf	279	3,272	1	12,180
ii32d3.cnf	824	19,478	1	1,200
ii32e3.cnf	330	5,020	1	3,900
par16-2-c.cnf	349	1,392	279	×
par16-4-c.cnf	324	1,292	10,390	×
par8-2-c.cnf	68	270	1	1
par8-4-c.cnf	67	266	1	12
ssa7552-158.cnf	1,363	3,034	×	×
ssa7552-159.cnf	1,363	3,032	×	×
ssa7552-160.cnf	1,391	3,126	×	175,500

使用マシン:SGI O2 180MHz 64 台

SASAT の使用した計算機とは, 1 台あたり 2.5 倍ほどの速度差があるが, それを考慮に入れても, 並列 GSAT が圧倒的に高速である. f800 例題のように解けなかった例題もあるが, それは GSAT と SASAT のアルゴリズムの能力の差であると考えられる. ssa 例題はローカルサーチでないプログラムでは高速に解かれており, 解くためには高速化よりも成功率向上の工夫が必要である.

5 おわりに

PVM による局所探索法の SAT アルゴリズムの高速化手法を提案した. この手法は非常に単純だが, 効果的に高速化が可能であることが確認できた. 今後は GSAT のアルゴリズムに改良を加え, さらに実用性を高める工夫などを行いたい.

参考文献

[1] A. Gaist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sanderam, "PVM:Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing", The MIT Press, 1994

[2] Bart Selman, Hector Levesque, Dabid Mitchell, "A New Method for Solving Hard Satisfiability Problems", *Proc. AAAI92*, 1992

[3] Bart Selman, Henry Kautz, "Local Search Strategies for Satisfiability Testing", *Second DIMACS Implementation Challenge Workshop*, 1993 nnn

[4] "Cliques, Coloring, and Satisfiability", *Second DIMACS Implementation Challenge*, 1993

[5] Elias Koutsoupias and Christos H. Papadimitriou, "On the greedy algorithm for satisfiability", *IPL43*, 53-55, 1992

[6] 河合大輔, 宮崎修一, 岡部寿男, 岩間一雄, "SAT に対する局所探索法のベクトル化", 電子情報通信学会 コンピュテーション研究会, Jan, 2000.