F-045

マルチコア環境に向けた高速並列SATソルバの開発

A Fast Parallel SAT Solver for Multi-Core Processor Environment

高見 明秀 † **鍋島 英知** † **岩沼 宏治** †
Akihide Takami Hidetomo Nabeshima Koji Iwanuma

1 はじめに

本論文では，実用的な高速並列 SAT ソルバの開発を目的として，最新の SAT ソルバ技術に基づくマルチコア環境向けの SAT ソルバを提案する．評価実験の結果，2 つの CPU 利用時に約 2 倍の速度向上が得られることを確認したので報告する．

命題論理の**充足可能性問題** (satisfiability problem; **SAT 問題**) は計算機科学や人工知能の基礎となる最重要問題の一つであり，プランニングやスケジューリング，有界モデル検査，暗号の検証など多くの実用的応用をもつ．この SAT 問題を解く **SAT ソルバ**は，近年劇的な性能向上を果たしており，Chaff[6] や MiniSat[3] などの最新のソルバは何百万もの節からなる SAT 問題を数分で解くことが可能である．

一方，近年複数の CPU コアを統合した**マルチコア CPU** を搭載した PC が急速な普及傾向にある．この背景には，消費電力や発熱量の問題からシングルコア単体での性能向上が難しくなってきていることが原因として挙げられる．従って，今後のソフトウェア開発においては，並列プログラミング技術を活用して複数の CPU を効率的に利用することが求められている．

本研究の目的は，マルチコア環境向けの実用的な高速並列 SAT ソルバを開発することにある．これまでにも複数の CPU を利用した SAT ソルバに関する研究 [7, 1, 4] はいくつかあるが，本研究では最新の技術に基づく実用的な高速並列 SAT ソルバを開発することを目的とする点が異なる．このために，SAT 2005 競技会で優勝したソルバである MiniSat をベースとして利用する．MiniSat は，冗長な探索領域を枝刈りするための矛盾節 (conflict clause) の学習 [5] や単位節の高速発見法である two watched literals [6]，最新の変数選択ヒューリスティクスである VSIDS [6] など SAT ソルバにおける最新技術を実装したソルバである．

本研究では，MiniSat をベースとして利用し，探索戦略のパラメータを調整することで，各 CPU が異なる領域を探索するようにバイアスをかけ，さらに学習節を共有することで探索領域の効率的な枝刈りを図る．

提案手法に基づくソルバを実装し，SAT 2005 競技会におけるベンチマーク問題集で評価実験を行った結果，2 つの CPU 利用時に約 2 倍の速度向上が得られることを確認した．また，4 つの CPU 利用時には，これまで MiniSat で制限時間内に解くことのできなかった問題を 2 問解くことができた．

本論文の構成は以下の通りである．2 章では本研究のベースとなる MiniSAT を簡単に紹介する．3 章で提案手法を示し，4 章でその実験結果を述べる．そして 5 章で関連研究を紹介し，最後の 6 章で本研究のまとめと今後の課題について述べる．

† 山梨大学, University of Yamanashi

2 MiniSat

MiniSat は SAT 2005 競技会の 3 つの産業問題部門と 1 つの手製問題部門で優勝したソルバである[*1]．本章では，MiniSat の特徴について簡単に紹介する．

まず SAT 問題を定義する．SAT 問題は**連言標準形** (conjunctive normal form; **CNF**) で与えられる．CNF は，命題または命題の否定を表す**リテラル** (literal) の選言 ($\vee$) の連言 ($\wedge$) である． リテラルの選言を**節** (clause) と呼び，リテラルの集合で表す．例えば $\{P, Q, \neg R\}$ は節である．SAT 問題を解くとは，全ての節を充足する命題の真偽値割り当てを求めることである．

MiniSat の基本動作は，DPLL 手続き [2] に基づいている．以下に手続きの概要を示す：

(1) 現在の真偽値割り当てにおいて単位節となる節（つまり，1 つのリテラルを除く他のすべてのリテラルに偽が割り当てられている節）が存在すれば，その節を充足するため，残っているリテラルに真を割り当てる．この単位節による真偽値割り当てを**単位節伝搬** (unit propagation) と呼ぶ．単位節伝搬により新たな単位節が生まれることがあるため，単位節がなくなるまでこの作業を繰り返す．もし真偽値割り当てに矛盾が生じた場合，後述する矛盾節の学習手続きを実行する．

(2) 単位節が存在しない場合は，ある変数選択ヒューリスティクスに基づいて，まだ真偽値が割り当てられていない変数を選択し，真または偽を割り当てる．これにより単位節が生まれることがある．その場合は (1) へ，そうでない場合は (2) を繰り返す．

真偽値割り当てに矛盾が生じた場合，その矛盾発生の原因を解析する [5]．例えば，矛盾の原因が $\neg P, Q, \neg R$ であった場合（つまり $\neg P, Q, \neg R$ に真を割り当てたことが原因で矛盾が発生した場合），MiniSat は符号を反転した節 $\{P, \neg Q, R\}$（これを**矛盾節** (conflict clause) または**学習節** (learnt clause) と呼ぶ）を生成し，SAT 問題に追加する．この節の追加によって，再び $\neg P, Q, \neg R$ に真を割り当てることを防ぐことが可能になる．つまり，同じ矛盾の発生を回避できる．さらにこの学習節が単位節になるまで真偽値割り当てをバックトラックし，(1) へ戻る．以上が MiniSat に基本動作の概要である．

以下では，MiniSat の変数選択ヒューリスティクスと，学習節の削減，およびリスタートについて述べる．

2.1 変数選択ヒューリスティクス

MiniSat の変数選択ヒューリスティクスは VSIDS (variable state independent decaying sum) [6] を基にしている．すべての変数は**活性度** (activity) と呼ばれる値

[*1] 正確には，MiniSat に SAT 問題を簡単化するフロントエンドを備えた SatELiteGTI が優勝している．

を保持しており，この活性度が最も高く，かつ，真偽値が未割り当ての変数から順に選択される．そして選択した変数に偽を割り当てる．

通常の DPLL 手続きでは，変数選択ヒューリスティクスにより選択した変数に真または偽を割り当てる．それにより矛盾が発生した場合には，矛盾を解消するために，その変数の真偽値を反転させる．もし反転済みの場合は，さらに直前の変数選択までバックトラックする．一方 MiniSat では，ヒューリスティクスにより選択された変数には単に偽を割り当てるだけである．矛盾が発生するたびに学習節が SAT 問題に追加されるため，もし必要があれば，単位節伝搬によって選択された変数に真が割り当てられることになる．

MiniSat では，矛盾の原因解析において，矛盾の発生に関係した変数の活性度を向上させる．矛盾発生に関わった変数は重点的に選択されることになり，局所的な探索を促す．これにより，その局所的な領域における矛盾解消が促進されることになる．また，ある変数の活性度が特定の閾値を超えたとき，すべての変数の活性度を定数で割り，減少させる．これによって，最近発生した矛盾に関係する変数が重点的に選択されるようになる．

また，一定の確率で，変数の活性度に関係なく，ランダムに変数を選択する．デフォルトでは 2% の確率でランダムに変数を選択する．

2.2 学習節の削減

変数の活性度と同様に，学習節も活性度を持つ．MiniSat は矛盾の原因解析において，ある学習節が矛盾の発生に関係している場合に，その学習節の活性度を向上させている．

学習節は，同じ矛盾の発生を避けるための有用な知識ではあるが，矛盾が発生するたびに生成されるため，その数は膨大な量になる．これはメモリの使用量や単位節発見のコストを増大させてしまう．そこで定期的に活性度の低い学習節を除去することで，矛盾の発生に寄与しない学習節を取り除いている．

2.3 リスタート

MiniSat では，SAT 問題の充足可能性の判定に役立たない領域の探索に陥ることを防ぐため，矛盾の発生回数がある閾値を超えるたびにリスタートを行い，そのような領域からの脱出を図る．**リスタート**とは，変数の真偽値割り当てをすべてクリアし，再度探索を実行し直すことをいう．ただし変数と学習節の活性度に変更はなく，また生成された学習節も保持したままである．従って，リスタートの前後において探索の手順は異なってくる．

3 高速並列 SAT ソルバ

本研究で提案するマルチコア環境向けの並列 SAT ソルバの構成を図 1 に示す．本システムでは，MiniSat をベースとした SAT ソルバを CPU の数だけ並列に実行する．各 CPU プロセスは同じ SAT 問題を解くが，探索戦略のパラメータを調整することで，異なる領域を優先的に探索するようにバイアスをかける．図中の探索領域の濃淡はこのことを示している．また各 CPU プロセスは，自身が学習した節を共有メモリ上に実装したデータベースに登録し，他のプロセスが学習した節を共有する．これにより探索領域の効率的な枝刈りを図る．そしてどれか 1 つのプロセスが充足可能または不可能を出力したとき，他のプロセスを停止させ，システムは終了する．

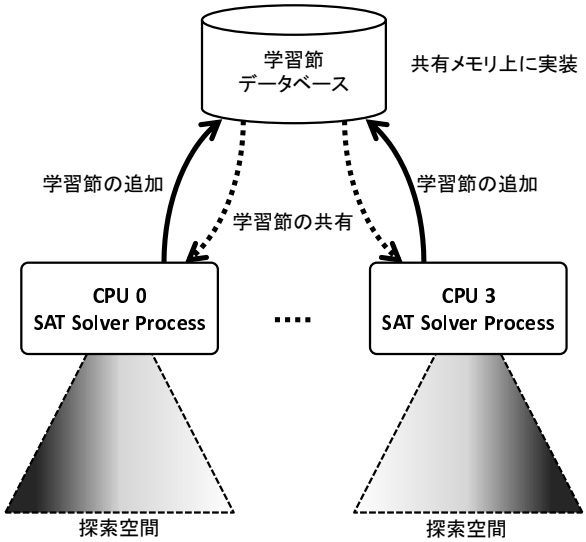

図 1　マルチコア環境向け高速並列 SAT ソルバの構成図

本章では，まず探索戦略のパラメータについて述べ，その後学習節の共有方法について述べる．

3.1 探索戦略のパラメータ

本研究では，探索戦略に関する 4 種類のパラメータを調整することで，各 CPU プロセスが異なる探索領域を優先的に探索するようにバイアスをかける．4 種類のうち 3 種類のパラメータはもともと MiniSat が実装しているパラメータである．

MiniSat には，変数及び学習節の活性度の向上の度合いを制御する 2 種類のパラメータと，ランダムな変数選択に関する 1 種類のパラメータがある．

(1) `clause_decay`: このパラメータは，学習節の活性度の増分を制御する．値域は $0.0 < \texttt{clause\_decay} \leq 1.0$ である．学習節の活性度は，その学習節が矛盾の発生に関わるたびに増加する．その増分を `cla_inc` とする．`cla_inc` は，リスタートが起きるたびに，$1/\texttt{clause\_decay}$ 倍される．すなわち，徐々に活性度の増分 `cla_inc` が増大していく．`clause_decay` が 1.0 に近いほど増分は緩やかに増加していく．学習節は定期的に削減されるが，`clause_decay` が大きいほど，矛盾発生に寄与しない節が積極的に削除される傾向にある．

(2) `var_decay`: このパラメータは，変数の活性度の増分を制御する．値域は $0.0 < \texttt{var\_decay} \leq 1.0$ である．変数の活性度も，その変数が矛盾の発生に関わるたびに増加する．その増分を `var_inc` とする．`var_inc` は，リスタートが起きるたびに，$1/\texttt{var\_decay}$ 倍される．すなわち，徐々に活性度の増分 `var_inc` が増大していく．`var_decay` が 1.0 に近いほど増分は緩やかに増加していく．探索が進むにつれ変数の活性度の増分が増大するため，最近矛盾の発生に関係した変数が選択されやすくなる．

(3) `random_var_freq`: このパラメータは，変数選択ヒューリスティクスにおいてランダムに変数を選択する確率を表している．

次のパラメータは，我々が新たに実装したものである．

(4) flip_order: MiniSat では変数選択ヒューリスティクス VSIDS により選択された変数に対し，偽を割り当てる．パラメータ flip_order はこれを制御するもので，flip_order が真の場合は，デフォルトの動作（つまり偽を割り当てる）を行い，flip_order が偽の場合は，選択された変数に真を割り当てる．

パラメータ flip_order は，選択された変数の真偽値の仮定の順序を入れ替えるため，探索の手順を大きく変更する効果があるといえる．

3.2 学習節の共有

本システムでは，MiniSat をベースとした SAT ソルバを CPU の数だけ並列に実行する．各 CPU プロセスは，リスタートが発生するたび，自身が学習した節を共有メモリ上に実装した学習節データベースに登録し，他のプロセスにより登録された学習節を読み込む（図 1）．リスタートの発生回数は SAT 問題に依存するが，通常数十回発生する．そのたびに学習節の登録と共有を行う．

学習節の長さは，短ければ短いほど探索領域を枝刈りする効果が高い．しかし問題によっては，ときに数百のリテラルからなる学習節を生成することがある．このような長い学習節は矛盾の発生に寄与する可能性が低いため，本研究では長さ 10 以下の学習節のみ登録・共有することにした．これは予備的調査の結果，共有する学習節の長さを 10 に制限した場合に最も良い性能を示したためである．

4 実験結果と考察

提案した並列 SAT ソルバを C++ 言語により実装し，評価実験を行った．評価用の問題として，SAT 2005 競技会で使用されたベンチマーク問題集から 48 問を選択した．SAT 2005 競技会では 1 問あたりの制限時間を 20 分としている．そこで問題の選択においては，少なくとも十数秒〜20 分程度かかる問題を選択した[*2]．問題には充足可能なものと不可能なものの両方が含まれる．また，オリジナルの MiniSat が 20 分以内に解けなかった問題を 2 問含んでいる．評価用 PC として，4 つの CPU を搭載する Mac Pro (CPU: Dual Core Intel Xeon 2.66GHz * 2, RAM: 4GB) を使用した．

探索戦略のパラメータを表 1 に示す．表中の V, C, R, F はそれぞれ clause_decay，var_decay，random_var_freq，flip_order を意味する．実験では使用する CPU の数を 1, 2, 4 と変化させて評価した．CPU 数が 1 の場合はパラメータ CPU0 を，2 の場合は CPU0 と CPU1 を使用した．CPU 数が 4 の場合は，すべてのパラメータを使用した．また表中の太字は，オリジナルの MiniSat とは異なるパラメータであることを表す．flip_order を除くと，clause_decay のみがデフォルトのパラメータと異なっている．これはパラメータ調整のために，オリジナルの MiniSat のパラメータを幾通りも試して予備実験を行った結果，clause_decay = 0.9999 としたときに，デフォルトのパラメータよりも多くの問題を解くことができたため採用した．

評価実験の結果を表 2 に示す．実験では，CPU の数が 1, 2, 4 の場合，かつ，学習節の共有あり／なしの場合について評価を行った．表中の解決数は，制限時間 20 分以内に解くことができた問題の数である．また時間は，すべての手法で共通して解くことのできた 46 問の解決時間の合計（分）である．向上率は，CPU 数が 1 の場合と比較したときの速度向上の割合である．なお，CPU 数が 1 の場合がオリジナルの MiniSat に相当する．

表 1　実行パラメータ

	V	C	R	F
CPU 0	0.95	0.999	0.02	true
CPU 1	0.95	0.999	0.02	**false**
CPU 2	0.95	**0.9999**	0.02	true
CPU 3	0.95	**0.9999**	0.02	**false**

表 2　実験結果

CPU 数	1	2		4	
学習節共有	なし	なし	あり	なし	あり
解決数	46	46	47	46	48
時間 [min]	83	60	44	54	34
向上率	-	1.38	1.89	1.55	2.47

また，特定の時間内に解けた問題数を表すグラフを図 2 に示す．縦軸が問題を解くのに必要な時間（秒），横軸がその時間内に解けた問題数を表す．分かりやすくするため縦軸の最大値は 500 秒に制限している．例えば，CPU が 4 つの場合で学習節を共有したとき，50 秒以内に解ける問題が 37 問あったことを示している．

実験結果より，CPU 数が 2 の場合には，学習節の共有なしでも約 1.4 倍の速度向上が得られ，さらに学習節を共有することで約 2 倍の速度向上効果があり，台数効果が得られていることがわかる．これは flip_order の変更により，2 つの CPU が探索領域を効率よく分担できたことを表している．一方，CPU 数が 4 の場合は，学習節の共有ありでも，約 2.5 倍程度の速度向上にとどまっている．これは clause_decay の変更だけでは各 CPU が探索領域をうまく分担できなかったことを表している．ただし，MiniSat が 20 分以内に解くことのできなかった問題を 2 問とも解いており，複数の探索戦略パラメータと学習節共有が大きな効果を持つことを示している．また図 2 より，ほとんどの問題で速度向上が得られていることが分かる．

5 関連研究

複数の CPU もしくは計算機を利用して SAT 問題を解く手法はこれまでにもいくつか提案されている．GridSAT[1] は，Chaff[6] をベースとしたグリッド環境向けの並列 SAT ソルバであり，比較的低速な回線で結ばれた計算機群を利用して SAT 問題を解く．GridSAT では，本手法と同様に学習節の交換を行う．また複数の計算機で変数の真偽値割り当てを分担することにより，探索空間の分割を行う．PaSAT[7] は，本研究と同様，複数の CPU を搭載した PC で動作する並列 SAT ソルバである．ただし学習節の交換の効果を検証することを目的としており，変数選択ヒューリスティクスには比較的単純なものを使用している．また GridSAT と同様に真偽値割り当てに基づく SAT 問題の分割を行う．GridSAT

[*2] SAT 2005 競技会の問題集の中には，MiniSat が数秒で解くことのできる問題も数多く存在する．

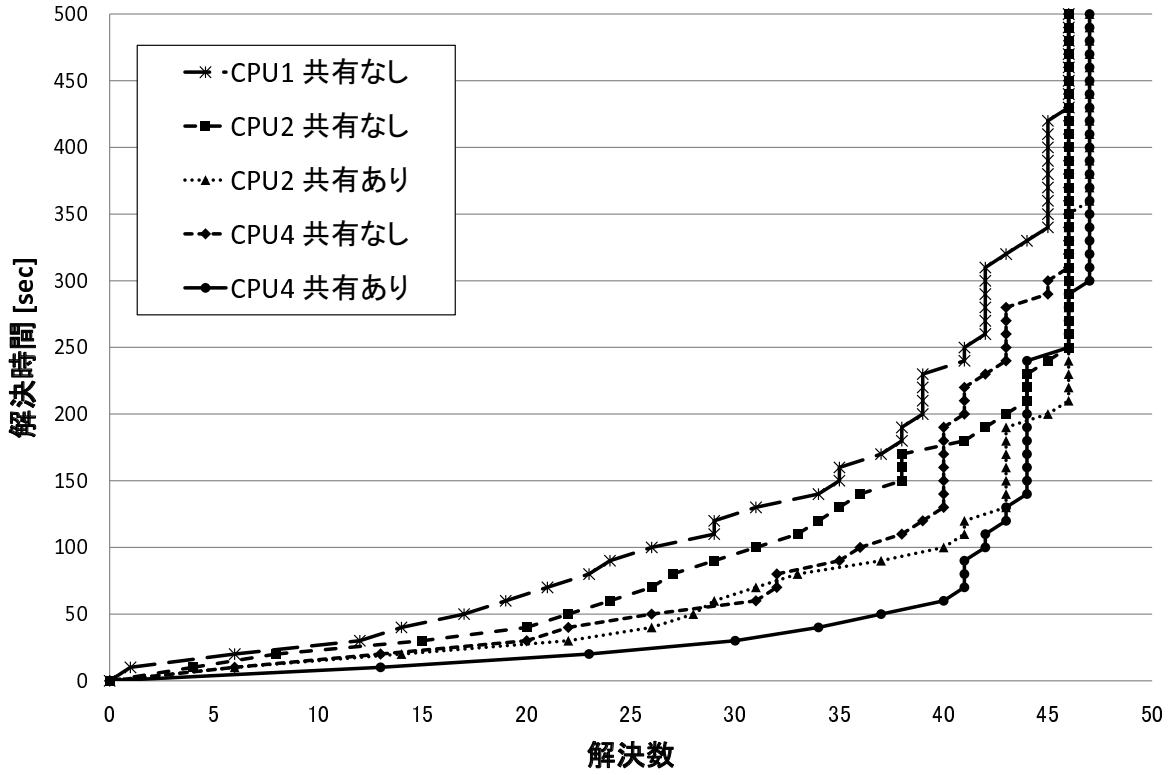

図 2　解決時間と解決問題数の関係

や PaSAT で用いられている探索空間の分担方法は本手法と異なる点である．

一般的に SAT ソルバには得手不得手な SAT 問題が存在する．Multisat[4] は，複数種類の SAT ソルバを並列実行することで，どのような問題にも強いソルバを構築することを目指しており，Java 言語で実装されている．ただし学習節の共有は一方的（つまり学習節を生成する機能を持つソルバが機能を持たないソルバに配布する）である．各ソルバが同じ SAT 問題を解く点は本手法と同様であるが，学習節の共有方法が異なる．また上記の 3 つの手法と比較して，本研究では実用的な高速並列 SAT ソルバを構築するため，最新の SAT ソルバ技術に基づいている点も異なる．

6　まとめと今後の課題

本研究では，マルチコア環境向けの高速並列 SAT ソルバを提案した．探索戦略パラメータの調整と学習節の交換により，単体実行と比較して大きく性能が向上することを示した．特に CPU 数が 2 の場合，約 2 倍の性能向上が得られており，`flip_order` の変更によって探索空間を効率良く分担できることを示している．

今後の課題には，GridSAT や PaSAT で用いられている SAT 問題の分割方法を併用し，さらなる速度向上を図ることや，Multisat のように異種のソルバを統合する手法の検討，探索戦略パラメータや補題交換のタイミングの検討などがある．また，SAT プランニングやスケジューリングなどの SAT を利用した問題解決手法では，本質的に複数の SAT 問題を解くことになるため，複数の SAT 問題をマルチコア環境・グリッド環境で効率よく解く手法の検討と実装も今後の重要な課題の 1 つである．

謝辞：本研究は一部，文部科学省科学研究費補助金 (No.19700135) の援助を受けている．

参考文献

[1] W. Chrabakh and R. Wolski. GridSAT: A chaff-based distributed sat solver for the grid. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 37. IEEE Computer Society, 2003.

[2] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[3] N. Eén and N. Sörensson. An extensible sat-solver. In *Proceedings of SAT-2003*, pages 502–518, 2003.

[4] K. Inoue, T. Soh, S. Ueda, Y. Sasaura, M. Banbara, and N. Tamura. A competitive and cooperative approach to propositional satisfiability. *Discrete Applied Mathematics*, 154:2291–2306, 2006.

[5] J. P. Marques-Silva and K. A. Sakallah. GRASP – A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48:506–521, 1999.

[6] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of DAC-01*, pages 530–535, 2001.

[7] C. Sinz, W. Blochinger, and W. Küchlin. PaSAT - parallel sat-checking with lemma exchange: Implementation and applications. In *Proceedings of SAT-2001*, pages 212–217, 2001.