2007－HPC－111 (6)
2007／8／1

SATソルバMiniSatの並列化とそのチューニング手法

大村 圭[†] 渋谷 健介[††]
稲垣 良一[††,☆] 上田 和紀[†††]

本論文では，充足可能性問題（SAT）の逐次ソルバであるMiniSatのlemma共有を用いた並列化とそのチューニング手法について述べる．SATソルバは特定の探索域には解がないということをlemmaと呼ばれるclauseとして学習する．このlemmaは探索木の枝刈りに非常に有効であるが，全てのlemmaを各PE間で共有するためには膨大な通信コストがかかってしまうため，効率の良い学習と通信が必要である．lemmaの渡し方や各パラメタをチューニングし，24PEのクラスタで実行したところ，逐次版では解けなかった問題のうち，27題を新たに解くことができた．

Parallelization of SAT Solver MiniSat and the tuning technique

Kei OHMURA ,[†] Kensuke SHIBUYA ,[††] Ryoichi INAGAKI [††,☆]
and Kazunori UEDA[†††]

This paper describes parallelization of the SAT solver MiniSat based on lemma sharing and its tuning techniques. When an SAT solver finds that a specific branch of the search space does not have a solution, it learns new clauses called lemmas. Lemmas are very useful for pruning search space. However, since sharing lemmas between PEs can take enormous communication cost, developing efficient learning and communication techniques is very important. Using parameters obtained by various experiments on lemma sharing, our parallel MiniSAT solved many more SAT competition problems than the original MiniSAT on a PC cluster with 24 PEs.

1. はじめに

命題論理式の充足可能性問題（propositional SATisfiability problem: SAT）は計算機科学の分野において古典的なNP完全問題として強い興味の対象となってきた．SATはNP完全という特性から多項式時間で解くことは一般にできないと考えられる．しかしソフトウェア・ハードウェア検証，AIプランニングといった様々な実アプリケーションが多項式時間でSATに還元可能であり，SATを高速に解く意義は大きい．

主流のSATソルバの基本となるアルゴリズムは，40年程前にDavis, Putnam, Logemann, Lovelandによって考案されたDPLLアルゴリズムであり，これに非時間順バックトラックと衝突の解析と学習機能を加えることによりSATソルバの性能は大幅に上がり，大規模な問題も解けるようになった[2]．

逐次SATソルバの性能を競う大会としてSAT Competition[9]，SAT-Race[8]が開催され，SATソルバの改良は続けられている．近年ではマルチコアやグリッド上で動く並列SATソルバも開発されているが，後述する問題により大幅な高速化は実現できていない．

本研究では，上述の大会で優秀な成績を修めた逐次ソルバであるMiniSat[7]をMPIを用いて並列化し，クラスタ上で実行することにより高速化を目指した．並列化の基本方針として，各PEにランダムに適当な探索域を探索させ，その際に学習した，特定の探索域には解がないことを示すlemmaを各PEで共有することにより，各PEが探索域を削減するよう実装した．また，lemmaの渡し方やMiniSatの各パラメタをチューニングすることにより，さらなる高速化を目指した．

2. SAT

SATとはある命題論理式が真になるような変数の割り当てを求めるか，またはどのような割り当てをしても真にならないことを示す問題であり，多くのアルゴリズムが考案されている．

SATはliteralを論理和で繋げたclauseをさらに論理積でつなげたCNF（Conjunctive Normal Form）を入力とする．どのような命題論理式も多項式時間内でCNF形式に変換可能であることが証明されている．

† 早稲田大学大学院 基幹理工学研究科 情報理工学専攻
†† 早稲田大学大学院 理工学研究科 情報・ネットワーク専攻
☆ 現在，特許庁
††† 早稲田大学 理工学術院

CNF の例

式 1 $= (a+b)(a'+b'+c)(a+b'+d)$
式 2 $= (a+b)(a+b')(a'+c)(a'+c')$

式 1 の場合，$a = T, b = F, c = F, d = T$ と真理値を割り当てることにより充足可能となる．また，式 2 の場合，変数をどのように割り当ててもなんらかの clause が偽となってしまうため，充足不可能な問題である．

3. 探索アルゴリズム

SAT を解くためのアルゴリズムは大きく分けて完全アルゴリズムと確率的アルゴリズムの二つに分類できる．確率的アルゴリズムは解の充足不可能を示すことはできないが，AI プランニングなどの充足不可能であることを示す必要のない問題に対して，高速に解を発見できる．しかし，検証をはじめとした多くの問題は充足不可能であることを示す必要があるため，完全アルゴリズムの需要は高い．

3.1 DPLL

DPLL に基づく SAT ソルバは大きく分けて変数選択，推論，衝突の解析と学習，バックトラックの 4 つのフェーズからなる．

3.2 変数選択

あるヒューリスティクスに従って真理値が未割り当ての変数を一つ選び，真か偽を割り当てる．同じ問題に対してもヒューリスティクスが異なると探索域の大きさが異なってくるが，MiniSat では VSIDS (Variable State Independent Decaying Sum) を採用している．VSIDS では各変数に対してカウンタを用意し，clause に出現する変数の総数を初期値とし，学習した lemma 中に出現する変数の総数を加算していく．また，カウンタの値は定期的に減らす．VSIDS は良い選択をするにもかかわらず，全体実行時間に対して VSIDS の実行時間は少ない．

3.3 推　論

変数選択を行った際に連鎖的に割り当てが決まる変数を探し出し，真理値を割り当てる．最も効果の高い推論技法が unit clause rule である．ある clause において一つの literal を除いた全ての literal に偽が割り当てられた場合，clause を真にするためには残る一つの literal に真を割り当てなければならない．先ほどの式 1 を例にすると，$a = F$ と割り当てた場合，$(a+b)$ を真とするために，$b = T$ と連鎖的に割り当てることができる．この一連の作業を Boolean Constraint Propagation (BCP) と呼ぶ．

3.4 衝突の解析と学習

ある変数の割り当てが T かつ F と推論されることを衝突という．衝突が発生した場合，ソルバはバックトラックをして変数割り当てのやり直しをする必要がある．その際，衝突の原因を解析して lemma と呼ばれる clause を追加して，以降の探索の高速化を図る．

lemma は探索域の削減につながる可能性を持つが，論理的には冗長な clause であるため，探索域の削減と clause の増加による処理時間の増加とはトレードオフの関係となる．そのため，効率的な lemma の学習が必要である．

衝突の学習方法は図 1 のような推論の連鎖を表す implication graph におけるグラフカット問題に相当する．図 1 において各変数の括弧の中の数字は，探索レベルを表している．探索レベルが 5 の変数がいくつか見られるが，それらは V11 の割り当てによって連鎖的に割り当てをされた変数である．変数選択により決定された変数から衝突を起こした変数までに至るパスが必ず通る変数を UIP (Unique Implication Points) という．UIP の中でも特に衝突を起こした変数に最も近い UIP を FirstUIP という．図において FirstUIP でのカットは $-V_{10}$, V_8, $-V_{17}$, V_{19} となるので，その否定を取った $(V_{10} + V_8' + V_{18} + V_{19}')$ を lemma として学習する．カット方法としては，FirstUIP 以外に，二つ目の UIP，全ての UIP でのカットといった方法があるが，一般的に FirstUIP でのカットによる lemma の追加が最もよい性能を示している[4]．

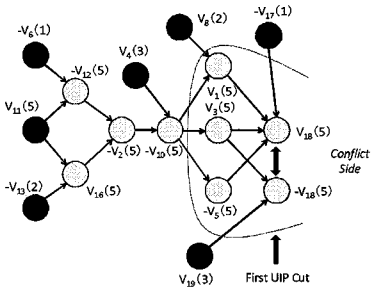

図 1 Implication graph[4]
Fig. 1 Implication graph

4. 並列 MiniSat の設計と実装

本研究では SAT ソルバ MiniSat をベースとしてクラスタ向きの並列化方式を検討する．

4.1 探索域の分割

SAT ソルバを並列化する際に問題となるのが，探索域の分割である．SAT の真理値の割り当ては BCP により行われることが多いため，探索域の大きさが予測できず，探索域を静的に分割することができない．

この問題に対して，各 PE の探索域が重複しないように動的に探索域を分割していく方法と，各 PE に最初の数回ランダムに適当な探索域を与え頻繁に lemma を交換することにより探索域の重複を最小限に留める方法とが考えられる．前者の方法では，未割り当て

の変数に対し，各 PE で真と偽をそれぞれ割り当てるため，各 PE の探索域は重複しないが，MiniSat は非時間順バックトラックを行うため，一回のバックトラックで浅いレベルまで戻ることが多く，均等に負荷分散させることは難しい．また，各 PE が異なる探索域を分担する場合も，学習した lemma は全 PE に共通のものであるため，共有を図らないと各 PE はもう探さなくてよい空間を探索してしまう可能性がある．一方，後者の方法では，他のプロセスの変数割り当てに関係なく各プロセスは探索を進めることができるが，ある程度の探索域の重複は避けることができない．以上からどちらの方法を取るにしても，探索域を重複なく，かつほぼ均等に分割するのは難しいことがわかる．本研究では他のプロセスの探索状況を気にすることなく探索を進められる後者の方法を取ることにした．

MiniSat の変数選択は VSIDS のヒューリスティックに従っている．そのため，初めの変数選択の時のみランダムに変数を選択してもその後の変数選択は重複してしまう可能性が高い．そこで，各 PE に対して，ランダムに変数を選択させる回数を変化させることにした．具体的には PE は自らのプロセスランク回だけランダムに変数を選択し，その後は VSIDS に従うように実装した．

4.2 lemma 共有

各 PE が解の探索の途中で学習した lemma をお互いに通信しあうことにより，探索域削減のための情報を共有することができる．各 PE の探索域をできるだけ削減するためには，lemma を頻繁に送受信しなければならないが，通信コストが膨大になってしまうため，通信のタイミングと送信する lemma の量は制限しなければならない．それ故，通信コストを考慮しながら，適切なタイミングと方法で lemma の受け渡しをする必要がある．

4.2.1 マスタ・ワーカモデル

lemma の管理と配布を行う方法として，我々はマスタ・ワーカモデルを採用した．一つのマスタが lemma の管理を行い，複数のワーカが探索を進める．

マスタはワーカから lemma を受信し保持する．また，その際に他ワーカから受信して保持していた lemma を通信中のワーカに送信する．全てのワーカに送信した lemma は適宜削除していく．

ワーカはそれぞれランダムに選んだ探索域から探索を始め，ある一定回数衝突を繰り返したらワーカと通信をして lemma の送受信をする．あるワーカが解を発見したら探索を終える．

4.2.2 階層構造

マスタ・ワーカの場合，マスタ 1 台に対しワーカが複数台となるため，あるワーカがマスタと通信中の場合，他のワーカは通信待ちとなる．そのため，PE 数を増やしていくと，通信にかかる時間の割合が膨大になってしまう．また，送信する lemma の量を増やした場合も，通信時間が増加してしまう．予備実験として取ったデータでは，PE 数や送信する lemma の量を増加させると通信時間にかかる割合が全体の実行時間の 9 割以上を占める結果になった．そこで，通信コストを削減し，PE 数や送信する lemma の量を柔軟に変更できるようにするため，マスタを管理するマスタを作成し親マスタ - 複数の子マスタ - 複数のワーカという階層構造にすることにより，一つの子マスタが通信の相手をするワーカを制限して，通信時間を削減することにした．

5. チューニングおよび評価

並列化実験に用いた環境は，1000 BASE-T Ethernet で接続された Dual core AMD Opteron 2.0GHz 2 個を搭載した PC9 台で構成され，OS は Cent OS 4.4，主記憶容量は 1 台当り 4GB，実行環境は SCore 6.0.2 である．

問題は SAT-Race 2006[8] で使用された問題を使い，5 回実行した結果の平均を測定値とした．各パラメタをチューニングする際に，並列効果がよく出ている問題をいくつかピックアップし，問題を解かせ，結果に一般性が見られるものをグラフや表として取り上げている．また，実行時間から通信時間を差し引いたものを探索時間として扱っている．以降は特に断りがない限り，親マスタ 1 台 - 子マスタ 5 台 - ワーカ 18 台の階層構造とし 8×3 台（24PEs）で実行している．

5.1 lemma による性能向上

実際に lemma を交換することにより性能が向上しているか評価するために，同じ変数選択方式と乱数で実行する PE で，lemma の交換をした場合としない場合の性能を比較した．純粋に lemma の効果を確かめるため，通信にかかった時間は差し引いて評価した．3×3 台（9PEs）で実行し，PE0 がマスタ，PE1 〜PE8 をワーカとし，$(2n-1)$ と $2n$ が同じ変数選択方式と乱数で実行するが，lemma 交換を行うのは奇数ランクのみとし 20 題解かせた．lemma を交換した場合の性能向上比の平均値を表 1 に示す．表から他 PE と lemma を交換することにより速度が向上していることがわかる．また，lemma を交換しなかった PE は問題を解けなかったケースもいくつか見られた．

表 1 lemma 交換による PE 毎の性能向上比
Table 1 The effect of lemma exchange on each PE

PE	PE1/PE2	PE3/PE4	PE5/PE6	PE7/PE8
ratio	2.12	1.85	1.55	2.23

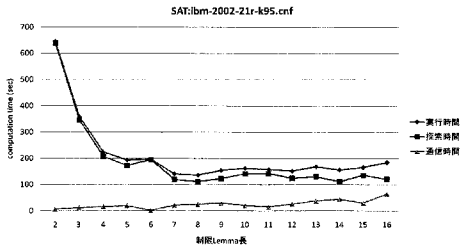

図 2 制限 lemma 長と実行時間（SAT）
Fig. 2 Limit lemma length and execute time (SAT)

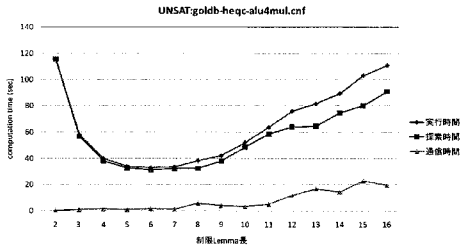

図 3 制限 lemma 長と実行時間（UNSAT）
Fig. 3 Limit lemma length and execute time (UNSAT)

5.2 制限 lemma 長

lemma は探索域の削減に寄与するが，論理的には冗長な clause である．また，lemma を多く交換しようとすると，通信量が膨大になるため，交換する lemma 量はある程度制限しなければならない．つまり，探索域の削減に役に立つ良い lemma のみを交換したい．一般的に良い lemma とは clause 長の短い lemma と言われている．今回は制限 lemma 長の値を変更してベンチマークを取ることで，並列実行の際に最適な lemma 長を探すこととした．

制限 lemma 長と実行時間の関係の例を充足可能（SAT）な問題と充足不可能（UNSAT）な問題について図 2，3 に示す．

問題によって最適な制限 lemma 長が異なるため，一概にどの値が最適であるというのは難しいが，大まかな傾向は掴める．やはり制限 lemma 長の値を大きくしすぎると，実行時間が増大していくので，制限 lemma 長は 4〜10 程度が最適な値であるといえる．

5.3 restart のタイミング

MiniSat にはある一定回数の衝突を繰り返すと，変数の割り当てを全てやり直す restart という機能がある．restart を行う衝突回数を 100 回とし，restart するたびに，その値を 1.5 倍していくため，探索が進むに連れ restart の回数は少なくなっていく．

並列 MiniSat では学習する lemma の量や種類が異なるため，衝突回数も逐次の場合と比べ異なると考えられる．そこで，restart の係数 1.5 の値を変えて，並列 MiniSat にとって最適な値を探した．その例が

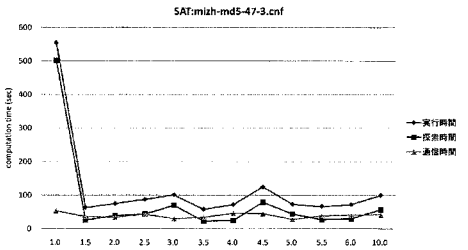

図 4 restart 間隔と実行時間（SAT）
Fig. 4 Restart interval and execute time (SAT)

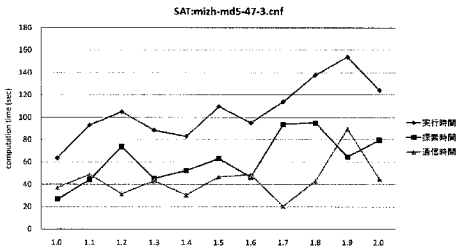

図 5 通信間隔と実行時間（SAT）
Fig. 5 Communication interval and execute time (SAT)

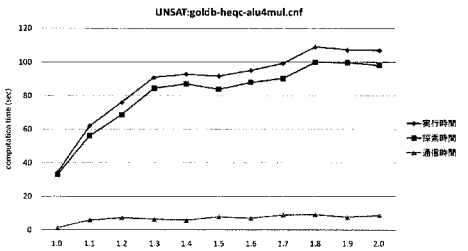

図 6 通信間隔と実行時間（UNSAT）
Fig. 6 Communication interval and execute time (UNSAT)

図 4 である．

図から係数が 1.0 の場合，特に遅くなっているのがわかる．係数を 1.0 とすると衝突を 100 回するたびに restart してしまい，変数割り当てが追い付かないためと考えられる．係数を大きくしていっても実行時間に大きな変化は見られなかった．

5.4 通信のタイミング

lemma の交換には多大な通信量がかかるため，学習と同時に lemma をマスタに送信することは効率的ではなく，通信のタイミングはある程度限定する必要がある．今回は前節で説明した restart と同様の方法を採用することにし，初期値を 100 としそれに係数 x を掛けていった．その例を図 5，6 に示す．

図からわかるように，SAT（充足可能）の問題も UNSAT（充足不可能）の問題も共に，係数が 1.0 に近いほど高速に解が求められていることがわかる．特

に UNSAT の問題に関しては，係数が大きくなるに従い，実行時間がほぼ単調に増加している．この結果より，lemma の交換は頻繁に行った方が良いことがわかる．lemma の交換を頻繁に行うことにより探索域の重複を最小限に留められるため，この結果は妥当であると思われる．

5.5 変数選択

4.1 節でも述べたように，本実装において，各 PE の変数選択は最初の数回をランダムに決定した後は，VSIDS に従っている．ここでは偶数ランクの PE のみ変数選択を完全にランダムにした場合，どのようになるかデータを取ってみたところ，ほとんどの問題は遅くなったが，2 倍ほど速度が向上した問題がいくつかあった．しかし，どの問題も解を見つけた PE は VSIDS を採用している奇数ランクである．このことからランダムに変数選択をすることにより速くなったのではなく，ランダムに変数選択をしている PE が学習した lemma が他の PE の探索域の削減に効果があったと考えられる．

5.6 逐次版と並列版の比較

SAT-Race 2006 の問題を 100 題，逐次版と並列版で解かせて，その結果を比較する．並列版の各パラメタは前節までに取ったデータを基に設定した．具体的には制限 lemma 長を 8，restart の係数を 1.5，通信タイミングの係数を 1.0，変数選択のヒューリスティックは基本的に VSIDS に従い，最初の数回はランダムに決定するよう各 PE 毎に設定した．探索は 1200 秒で打ち切り，それ以上かかる問題は解けなかった問題として扱った．

表 2 は各ソルバが解くことができた問題数である．問題の充足可能性にかかわらず，並列版は逐次版より多く問題を解くことができており，合計で 27 題多く解けている．表 3 は実行時間中の通信時間が占める割合を示している．SAT も UNSAT も平均して 3 割以下の割合となっている．

逐次版と比較した並列版の各問題毎の性能向上比を図 7，8 に示す．またその詳細の一部を表 4 に示す．図 7，8 では逐次版での実行時間が小さかったものを左にして並べており，元々速く解けていた問題は並列化しても通信コストにより逆に遅くなる可能性が高いが，図よりほとんどの問題では性能向上していることがわかる．問題によっては 23.96 倍と，台数以上の効果がでているものもある．

また，並列版のみ解けた問題の実行時間を図 9 に示す．並列版で解けた問題の中での最短実行時間は 4.04 秒（SAT）および 52.75 秒（UNSAT）で，これはそれぞれ少なくとも 297 倍，22 倍の性能向上が得られたことを示している．

SAT と比べて UNSAT の方が性能向上比が低いのは，UNSAT は全ての探索域に解がないことを示さなければならなく，SAT の問題と比べて探索域が重複しやすく，並列効果が出にくいためだと考えられる．

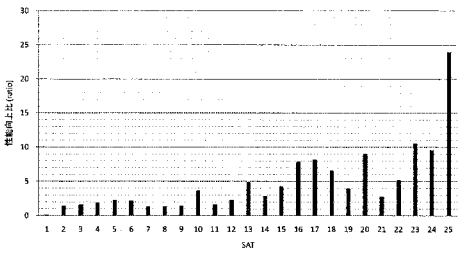

図 7 並列 MiniSat 性能向上比（SAT）
Fig. 7 Speedup ratio of parallel MiniSat (SAT)

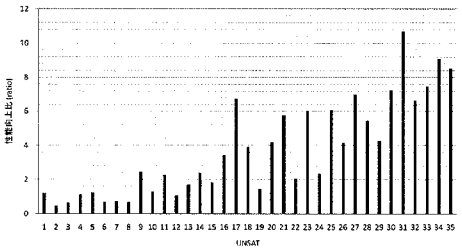

図 8 並列 MiniSat 性能向上比（UNSAT）
Fig. 8 Speedup ratio of parallel MiniSat (UNSAT)

表 2 解いた問題数
Table 2 The number of solved problems

Problem	SAT	UNSAT	Total
逐次版	25	35	60
並列版	40	47	87

表 3 総実行時間中の通信時間の割合
Table 3 The ratio of communication time in the total execution

Problem	SAT	UNSAT
割合	0.29	0.23

6. 関連研究

並列 SAT ソルバは本研究以外にもいくつか提案されている．そのうちのいくつかをここで紹介する．

6.1 MiraXT[5)]

zChaff をベースとしている SAT ソルバ Mira をマルチスレッド化し，マルチコア CPU 上で並列化している．各スレッドはある変数に対して他のスレッドとは違う真理値を割り当てることにより動的に探索域を分割している．また，共有データベースに問題とそれぞれのスレッドが学習した lemma を保持している．スレッドはデータベースにアクセスする際にロックを掛けて他スレッドがアクセスできないようにし，データベースの整合性を保っている．industrial の多

表 4　実行時間（sec）と性能向上比
Table 4　Execute time and the performance enhancement ratio

Problem	SAT1	SAT20	SAT25	SAT26	UNSAT7	UNSAT28	UNSAT31	UNSAT36
逐次版	0.30	424.31	876.45	>1200	3.23	190.49	401.49	>1200
並列版	4.32	46.93	36.57	4.04	4.47	34.90	37.64	52.75
ratio	0.07	9.04	23.96	>297	0.72	5.46	10.67	>22

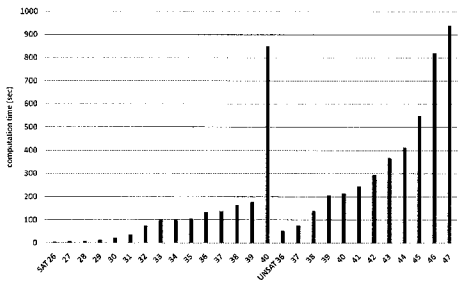

図 9　並列版のみ解けた問題の実行時間
Fig. 9　The number of solved problem by parallel MiniSat

くの問題に対して速度向上が見られた．

6.2　multisat[6)]

異なる SAT ソルバを PrologCafe をプラットフォームとしてマルチスレッド化することで並列実行することができる．異なる SAT ソルバを並列化してそれぞれの苦手分野を補うことによりどのような問題に対しても安定した結果を出すことをめざしている．Java で記述された SAT ソルバならインターフェースをあわせるだけで並列実行可能である．

7.　まとめと今後の課題

本論文では，SAT ソルバ MiniSat をクラスタ上で並列化し，lemma の渡し方や各パラメタをチューニングする手法を提案した．lemma の渡し方やタイミング，量などを最適化することにより一定の性能向上が見られた．特に SAT の問題に対しては，問題によっては大幅な性能向上が見られた．SAT の問題は探索木のどの部分に解があるかわからないため，できるだけ様々な探索木を多くの PE で解かせることによる効果が大きいためだと考えられる．逆に UNSAT の問題は充足不可能であることを示さなければならないため，ある程度探索域が重複してしてまう本実装では，小規模な問題に対しては効果が出にくかった．

今回の実験により，通信コストを多少犠牲にしても頻繁に lemma を交換するのは有効であることが分かった．しかし学習した lemma のうち，実際に役に立つものはわずかであるとも言われている．交換した lemma のうち，実際にどの lemma が役に立っているかを調べることにより，交換する lemma を限定できれば，さらに通信コストを削減できると考えられる．

本実装では，元の MiniSat の性能を損なうことのないよう，MiniSat のパラメタはほとんど変えていない．しかし並列版では lemma を交換するため，ソルバの探索時の挙動は逐次版とは多少変わっていると考えられる．そのため，交換した lemma がソルバの処理にどのような影響を与えているか調べ，逐次向けの各パラメタを並列向けにチューニングする必要がある．

5.5 節で奇数ランクと，偶数ランクで変数選択の方式を変更したところ，問題によっては速く解くことができた．つまり，変数選択が異なると，学習する lemma も異なってくると考えられる．しかし，問題を解いた PE はいずれも変数選択の方式が VSIDS であり，変数選択の方式をランダムにした PE は lemma を交換することのみを目的として，問題を解くことは捨てた PE になってしまう．今後，各 PE はどのような lemma を学習しているかなどを詳しく調査して，さらに効率のよい学習をできるようにする必要がある．

参　考　文　献

1) 大村 圭: “SAT ソルバ MiniSat の学習節共有を用いた並列化”, 2006 年度 卒業論文, 早稲田大学理工学部コンピュータ・ネットワーク工学科, 2006.
2) Lintao Zhang, Sharad Malik: “The Quest for Efficient Boolean Satisfiablity Solvers”, in Proc. CAV'02, LNCS 2404, Springer, pp.17-36, 2002.
3) Niklas Eén, Niklas Sörensson: “An Extensible SAT-solver”, in SAT 2003, LNCS 2919, Springer, pp.502-518, 2004.
4) Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, Sharad Malik: “Efficient Conflict Driven Learning in a Boolean Satisfiability Solver”, in Proc. ICCAD2001, pp.279-285, 2001.
5) Matthew Lewis, Tobias Schubert, Bernd Becker: “Multithreaded SAT Solving”, in 12th Asia and South Pacific Design Automation Conference, 2007.
6) 番原睦則, 田村直之, 井上克己: “Prolog から Java へのトランスレータ処理系とその応用”, 日本ソフトウェア科学会第 22 回大会（2005）.
7) MiniSat Page: `http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/Main.html`
8) SAT-Race 2006: `http://fmv.jku.at/sat-race-2006/`
9) SAT Competitions: `http://www.satcompetition.org/`