変数のアクティビティ情報を共有するマルチスレッドSATソルバ

A Parallel SAT Solver with Shared Variable-Activity

谷口 清則† 大森 晋作†† 長谷川 隆三††† 藤田 博††† 越村 三幸†††

†九州大学大学院システム情報科学府 ††NEC

††† 九州大学大学院システム情報科学研究院

1 はじめに

SAT(Boolean satisfiability problem)は，与えられたブール式を真とするような変数の値の割当てがあるかどうかを判定する問題であり，計算機科学における様々な問題がこれに還元可能なため，古くから重要視されている．

我々は，SATソルバminisat[1]をJava化したHerrsatのマルチスレッド化[2]に取り組んでいる．SATソルバでは動的に作られる学習節による探索空間の枝刈り効果が大きいことが分かっており，我々の実装でもスレッド間で動的に生成される学習節を共有している．

実験によるとこの実装では，複数回同じ問題を解くと実行時間のばらつきが大きく，数倍の差がでることもある．この原因として，並列に動作する各ソルバの変数の選択順が互いに独立していることが考えられる．選択された変数に基づいて，探索が分岐し，この選択順によって探索空間の大きさは変動するので，よい選択順をみつけることは重要である．

Herrsatにおいて，変数の選択順序は，VSIDSと呼ばれる手法に基づいている．VSIDSでは，各変数にアクティビティが付与され，アクティビティの大きい変数を選択する．アクティビティは，学習節に出現したときに増加する．

従来の実装では，アクティビティ情報をスレッド間で共有しておらず，各スレッド毎に独自のアクティビティ情報を持っていた．

本研究では，一つのソルバのアクティビティ情報を全ソルバ間で共有し，各ソルバがそれを基にそれぞれ異なるアクティビティを各変数に与える．変数の選択順序に関連を持たせることで，従来の推論では得られない学習節の取得・共有を行い，実行時間の高速化を図る．

2 Herrsatのマルチスレッド化

Herrsatでは，一定回数以上の矛盾が発生したときに，一度推論を破棄し，推論を最初からやり直すrestart戦略を実装している．restart後も，学習節やアクティビティ情報を保持しているので，より効率の良い推論が期待できる．

基準となる矛盾の回数の初期値は100で，以後restartのたびに1.5倍ずつ増やす．

2.1 マルチアクティビティによる並列化

Herrsatでは，推論木の分割ではなく，変数の選択順序が異なる複数の推論木間で，学習節の共有を行っている．これにより，シングルスレッドでは，自力で獲得できなかった学習節を獲得することができ，お互いの探索空間の削減し，高速化を行っている．

この実装では，推論開始時のアクティビティ情報は全ソルバで同一であり，変数の順序を決めるもう一つの要素である乱数の種のみが異なっている．変数の選択は主にアクティビティ情報に基づいて行われるが，一定の確率でランダムにも行われる．このとき，乱数が使われる．

しかし，推論中に，アクティビティ情報を共有しておらず，変数の選択順序が独立しており，実行結果のばらつきが大きい．これは，ソルバが協調して効率の良い探索を行えていないからと考えられる．

2.2 アクティビティ情報の共有による並列化

上記並列化に対し，本稿では，アクティビティ情報を共有する並列化手法を提案する．本手法では，解を探索するソルバ(以下，求解ソルバ)一つと，それを補助するソルバ(以下，補助ソルバ)を複数起動する．補助ソルバは，求解ソルバのアクティビティを基に求解ソルバが本来探索を行わないような変数選択を行い，有用な学習節を求解ソルバに渡す．

2.2.1 求解ソルバのアクティビティ情報送信

求解ソルバは，探索中に，ソルバの共有情報をもつ領域に，自身のアクティビティ情報をコピーする．コピーするタイミングは，学習節を生じた後であり，コピーする頻度は，restart間に3回行うようにしている．

2.2.2 補助ソルバのアクティビティ情報取得

補助ソルバは，restart直後に共有情報をもつ領域から，アクティビティ情報を取得する．その後，取得したアクティビティ情報の値を操作し，推論を行う．

本稿の実装では，取得したアクティビティ情報の

表 1: 実験結果

問題	変数の数	時間（秒）			矛盾数			conf id0/total		ANS
		1	2	new 2	1	2	new 2	2	new 2	
clauses-2.renamed	75528	4.98	3.97	5.57	5050	4560	5354	0.52	0.91	SAT
clauses-4.renamed	267767	139.33	58.07	93.81	71791	42708	44490	0.50	0.91	SAT
qg1-08	512	1.54	2.06	2.03	7997	10618	8056	0.50	0.56	SAT
qg3-09	729	7.55	5.64	6.04	38024	43790	43178	0.50	0.50	UNSAT
uf250-029	250	13.49	6.49	13.04	304423	242291	449596	0.50	0.50	SAT
uuf250-071	250	14.57	7.34	13.47	330229	275216	470861	0.50	0.50	UNSAT

値を逆数にすることで，求解ソルバの変数の選択順序を逆順にし，推論を行っている．

3 実験および評価

本節では，提案した手法の実験を行い，その結果についての評価を行う．

3.1 実験

今回の実験では，従来の手法でのシングルスレッドと 2 スレッド，提案した手法の 2 スレッドの実験を行い，比較を行った．2 スレッドは，いずれも 10 回の試行の平均を取っている．

実行環境は次の通り．

CPU: Intel Core 2 Duo E6600 (2.4GHz Dual Core)

Memory: DDR2 SDRAM PC6400 2GB

OS: Microsoft Windows XP Service Pack 2

Java VM: Java(TM) 2 Runtime Environment, Standard Edition (build 1.6.0_03-b05)

使用した問題に関しては，ベンチマークとして近年一般的となった Industrial，Random，Craft のカテゴリーから，2 問ずつ選んだ．

Industrial: 回路検証やネットワーク検証など現実の工業分野から生じた問題．(clause-2.renamed, clause-4.renamed)

Crafted: N-Queens，ラテン方陣等パズル系の問題．(qg1-08，qg3-09)

Random: ランダムに生成された問題．主に 3SAT．(uf250-029，uuf250-071)

実験結果は，表 1 に示す．表中の 1，2 は従来の手法でのシングルスレッドと 2 スレッド，new2 は提案手法の 2 スレッドである．conf id0/total は，矛盾回数における求解ソルバの割合を示す．この値が 0.5 であれば，求解ソルバと補助ソルバの仕事量は，ほぼ同じであると考えられる．また，1 に近づけば求解ソルバの仕事量が大きく，逆に 0 に近づけば補助ソルバの仕事量が大きいものとみなせる．

3.2 考察

実験の結果，シングルスレッドより高速になったものもあるが，従来の手法より低速であった．

原因として，補助ソルバの変数の選択順序を求解ソルバの逆順にする手法が，探索空間の枝刈りに貢献する学習節を生み出していない可能性が考えられる．

また，変数の数が多い問題では，conf id0/total の値が高く，補助ソルバがほとんど機能していないことがわかる．これは，アクティビティ情報の取得後の変数並べ替えの処理コストが高いためと考えられ，この処理の実装方法を変える必要がある．

4 まとめと課題

本稿では，アクティビティ情報を共有するマルチスレッド SAT ソルバの提案を行い，実験・評価を行った．実験の結果，従来手法より高速化することはできなかった．

課題としては，探索空間の枝刈りに高い効果を発揮する学習節を取得するようなアクティビティ情報の操作，および補助ソルバのアクティビティ情報取得後の操作のコストを小さくするような実装手法の考案である．

現在，3 スレッド以上に拡張し，他ソルバで上昇したアクティビティを減少させることで，ソルバ間で異なったアクティビティになるようにしていこうと考えている．

参考文献

[1] Niklas Eén, Niklas Sörensson. An Extensible SAT-solver [extended version 1.2], SAT2003.

[2] 大森 晋作. SAT ソルバ Herrsat の開発によるマルチスレッディング SAT ソルバの性能検証，九州大学大学院 システム情報科学府 修士論文，2007.