1M2-OS-02b-5

CDCL SAT ソルバにおける貪欲論理制約伝播

A Greedy Boolean Constraint Propagation in CDCL SAT Solvers

楢崎修二 *1
Shuji Narazaki

*1長崎大学
Nagasaki University

CDCL アルゴリズムは矛盾を検出した際に矛盾を解消する学習節を追加することで，効率のよい探索を行う SAT 問題のためのアルゴリズムである．本論文ではその改良案として，矛盾を検出しても直ちにバックジャンプするのではなく，複数の矛盾が起きるまで伝播を貪欲に続け，その後，複数の矛盾の解析結果に基づいてバックジャンプをするという手法を提案する．この利点は矛盾検出のコストの削減と，複数の矛盾を組み合わせることによるよりよい探索の実現である．実験により提案手法の評価を行うと，適切なパラメータを選択することにより伝播回数の 1 割以上の削減を得ることが出来たが，問題によらないパラメータの設定という課題があることがわかった．

1. はじめに

命題論理式の充足可能性を判定する SAT ソルバは近年の高速化より多くの分野へ応用されるようになった基礎技術の一つである [Biere09]．NP 完全性が最初に証明された探索問題ではあるが，矛盾の発生時に原因を解析し学習節として問題そのものに加える CDCL アルゴリズム，種々のデータ構造の開発，ヒューリスティクスの導入などにより実用を可能にする高速化が達成されている．現在もアルゴリズムの並列分散化やより高度なヒューリスティクスの導入など様々な発展的な研究がなされているがさらなる高速化が求められている．また，多くの研究は CDCL アルゴリズムの枠組みを変えるものではない．

本研究では CDCL アルゴリズムの変種として，矛盾が発生しても必要であれば探索を続けるという手法を提案する．以下，2 章において背景知識を，3 章で提案手法の詳細について説明する．4 章ではベンチマーク問題を使って妥当性についての評価を行い，5 章でまとめる．

2. CDCL アルゴリズム

まず基本用語を説明する．SAT 問題は与えられた命題論理式を充足させるように，含まれる変数（正負を区別する場合はリテラルと呼ぶ）に適切な値割当てを行うことである．変数の値は，それまでの割り当てによって必然的に決まる場合（これを**含意による割当て**と呼ぶ）と，一意に定めることが出来ないため仮に値を与える**決定による割当て**の 2 種類によって定まる．決定による割当てではその後矛盾が発生した場合はその値を取り消し，逆の値を与えることになる．決定によって値が与えられた変数の個数を**決定レベル**という．

図 1 に CDCL アルゴリズム [Silva99] での処理の推移を示す．一般的に，未割当てリテラルの一つ l_{i0} を決定により割当てることから処理が始まる．割当てを行ったことにより，式が矛盾していないか，または含意すべきリテラルが存在しないかを調べることが必要になるが，これは式を構成する節それぞれの状態を検査することで実現できる．この，節を対象とした検査を**論理制約伝播**と呼ぶ．多くの場合，決定による割当ての後には含意による割当てが複数回続く $(l_{i1}, l_{i2}, \cdots)$．含意できるものがなければ新たに未割当てリテラル l_{j0} を一つ選び決定する．決定したことで決定レベルは 1 上がる（従って $j = i + 1$ である）．この処理を未割当て変数がなくなるまで繰り返す．またはその途中である変数に真と偽の両方の値を割り当てる必要が出てきた場合，例えば $l_{kn} \wedge \overline{l_{kn}}$ が含意された場合には式は矛盾することになる．ここで CDCL アルゴリズムはどの割当て済みリテラル集合から矛盾が発生したかを解析し，その集合に含まれるリテラル (図では l_{j0}) を一つ選び，それ以降の割当てを全て破棄し（これを**バックジャンプ**という），逆の値 $\overline{l_{j0}}$ で（含意により）割り当て直す．さらに同じ矛盾を発生しないように，解析結果を論理式として与えられた式に追加する（正確には解析結果は節（**学習節**と呼ばれる）となり，学習節の集合に追加される）．

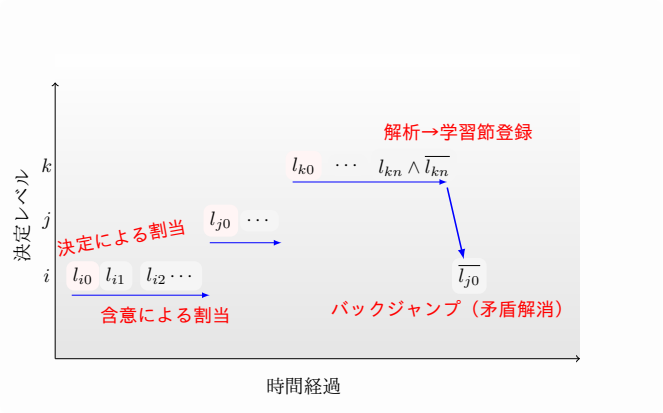

図 1: CDCL アルゴリズムにおける処理の時間推移

3. 提案手法：多矛盾バックジャンプ法

CDCL アルゴリズムでは矛盾が発生すると直ちに論理制約伝播をやめ，矛盾の解析とそれに引き続くバックジャンプを行う．従って，学習の回数と学習節の数，バックジャンプの回数，矛盾回数などはいずれも同一値となる *1．

ここで，矛盾によってもたらされる学習節とは問題から獲得された一種の知識である．より早い段階で，そしてより適切な知識を得ることはその後の探索の効率を左右すると考え

連絡先: 長崎大学大学院工学研究科，narazaki@nagasaki-u.ac.jp

*1 実際には学習節があまりにも増加するため，定期的な削減が行われるので最終的な学習節の個数は矛盾回数とは異なる．

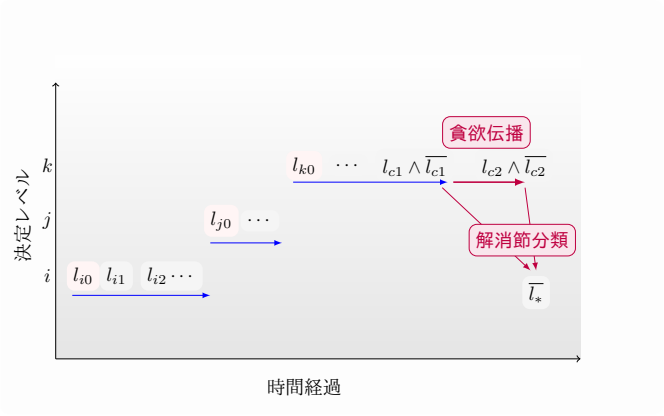

図 2: 多矛盾バックジャンプ法における処理の時間推移

られる．従って矛盾を多く効率よく発見することは一つの改善手法となりうる．そこで本論文では，バックジャンプのタイミングを遅らせ，複数の矛盾を発見する手法を提案する．概要を図 2 に示す．本手法では最初の矛盾 $l_{c1} \wedge \overline{l_{c2}}$ の検出後も論理制約伝播を進め，複数の矛盾 $l_{c2} \wedge \overline{l_{c2}}, l_{c3} \wedge \overline{l_{c3}}, \cdots$ を得る．これらの解析を行い，得られた知識（矛盾を解消する節）を総合して，バックジャンプレベルの決定や学習節の登録などを行う．提案手法を**多矛盾バックジャンプ法 (Multi-Conflict Backjump(MCB)**，その前半部である最初の矛盾で停止しない伝播処理を**貪欲伝播**，後半部の複数の矛盾からの解析→学習処理を**解消節分類**と呼ぶことにする．両者の具体的なアルゴリズムは次節以降にて説明する．

3.1 貪欲伝播に関する方針

この手法が有効であるためには複数の矛盾が存在すること，矛盾発生後にも論理制約伝播を続けられることが必要である．複数の矛盾が存在することは実験で確かめられる [楢崎 17] が，以下の点について検討する必要がある．

1. 矛盾発生後の伝播が論理的な間違いを起こさないこと
2. 加される伝播回数（付加される計算コスト）はそれによる効果（将来の伝播回数の削減量）を下回ること

前者は矛盾しているリテラルを含む節が単位節になり別のリテラルを含意する場合に問題となる．矛盾から割当ては発生してはならない．そこで，矛盾しているリテラルのため，従来は { 未割当，真，偽 } の 3 値で表される割当てに対し，「矛盾中」を意味する第 4 の値を追加する．この値を持つリテラルを含む節は充足しているものとすればよい．この値は矛盾発生後に割当て配列に現れ，バックジャンプ後には消滅するため，この値を見るのは論理制約伝播を行う関数からバックジャンプを実行する関数（Minisat では `propagate` から `cancelUntil` に対応）までとなり，限定された変更で実装可能である．

後者については，我々は追加伝播回数に上限を持たせることにした．上限は静的に得られる変数や問題中の節数などや，決定レベル・未割当変数数など実行中の動的状況に対応した値などから決めることになる．これまでの実験の結果では下式がもっともよい値を示した．

$$P^+ = p_1 \left(\frac{N}{P} \right)^{p_2} \tag{1}$$

ここで P は最後のバックジャンプから 1 つ目の矛盾検出までの論理制約伝播回数，N はその間に割り当てられた変数の個数，p_1, p_2 は制御パラメータである．最初の矛盾を検出した後，上式により計算される伝播上限 P^+ までの伝播を行う．直感的にはバックジャンプ以降に多くの変数が割り当てられた場合には効率よく複数の矛盾が検出できるだろうという見込みを反映したものである．

具体値は 4 章での実験結果で示すが，この式から与えられる追加伝播上限は決して大きなものではない．多くの問題で 1% 程度の確率で 2 つめの矛盾を検出する程度のものである．

3.2 解消節分類に関する方針

次に得られた矛盾を分析することで矛盾を解消する節（本来これは学習節と呼ばれているものであるが，本手法ではこの後の処理によって学習節データベースに登録されず（すなわち学習されず）に捨てられるものが存在するため，登録が確定していないこの時点の節を解消節と呼び分けることにする）を複数得る．これらから以下を決定するのが解消節分類である．

- バックジャンプレベル：どの決定レベルに戻るか
- 学習節集合へ登録すべき節の集合（解消節集合の部分集合）
- 含意対象リテラル集合：矛盾の原因となっており，ただちに割当て直されるべきリテラル

それぞれについて，以下ではこれまでの実験から得られた最もよいと思われる方針について説明する．

まずバックジャンプレベルについては，解析の結果得られた解消節 $c \in \boldsymbol{R}$ が示す最も小さな決定レベルとする．

$$l_{\downarrow R} = \min_{c \in \boldsymbol{R}} l_{\downarrow}(c) \tag{2}$$

これは検出された全ての矛盾を解消するためである．以下では $l_{\downarrow R}$ へのバックジャンプによって含意がされるリテラルを解消リテラルと呼ぶ（一つとは限らない）．

次に登録される節は節長（$|c|$ で節 c の節長を表す）を基に以下のように決定する．

$$|c| \leq \min_{c' \in \boldsymbol{R}} |c'| + p_3 \tag{3}$$

すなわち，最小節長を基準に制御パラメータ p_3 までの節 c を登録する．これは節長が大きく今後の学習節データベースでの削減の対象となりやすいものは最初から登録しないということを意味する [*2]．少なくとも 1 つの矛盾に対する節は学習されるため，いくつかの解消節を登録しなくても停止性には影響しない．選ばれた節の作る集合を $\boldsymbol{L}$ とする．

一方，選ばれなかった解消節節の持つ情報を有効利用するため，下式を満たす場合には，節から含意されるリテラル値を極性として登録することにした．リテラルの極性とは最後に割当てられた値（または直近の割当ての偏り）である．Minisat などでの変数 `phases` に対応する．

$$\forall c \notin L : l_{\downarrow}(c) \leq l_{\downarrow R} + p_4 \tag{4}$$

すなわち，選ばれたバックジャンプレベルに近いレベルを持つ解消節に対応する解消リテラルはその極性を反転し，次に決定による割当てがなされる際に矛盾を避ける値が選ばれる．バックジャンプレベルから遠いものは値の反転の効果が薄いと考えられるため制御パラメータ p_4 を上限とする．

*2 節のよさを示す尺度として LBD[Audemard09] も広く用いられているが，実験に用いたソルバは尺度として節長を用いていることに合わせた．

表 1: 3SAT($n = 250$) 問題 $\times 100$ での実行結果（平均値）

p_1	改善数	悪化数	$(p_2, p_3, p_4)\|p_1$			伝播回数	矛盾検出数	登録数	backjump
0.0	–	–	0.00	0	0	12109075.0	41776.5	41776.5	41776.5
1.0	0	512	0.10	1	1	12150851.5	41776.5	41776.5	41776.5
2.0	**86**	426	0.30	5	2	10513491.5	36989.9	36684.4	36465.6
3.0	33	479	0.40	1	4	10609980.7	37445.6	36997.9	36872.4
4.0	14	498	0.45	5	1	11532477.9	40790.7	40206.2	39834.1
5.0	36	476	0.15	2	6	10649122.1	38732.2	37103.2	36581.8
6.0	29	483	0.35	4	3	10760033.0	38724.8	37447.7	36861.8
7.0	20	492	0.45	6	1	10741357.6	38877.8	37611.9	36954.7

表 2: `38bits_10.dimacs.cnf` での実行結果

p_1	改善数	悪化数	$(p_2, p_3, p_4)\|p_1$			伝播回数	矛盾検出数	登録数	backjump
0.0	–	–	0.00	0	0	191761341	58632	58632	58632
1.0	0	512	0.10	1	1	191819973	58632	58632	58632
2.0	29	483	0.10	4	1	89525602	41421	41168	41086
3.0	16	496	0.30	1	1	26923701	16436	16342	16325
4.0	33	479	0.10	3	2	54000170	20184	19979	19940
5.0	76	436	0.35	5	2	6621410	4319	4287	4274
6.0	**113**	399	0.35	2	3	2134262	1736	1714	1707
7.0	66	446	0.15	2	1	1144671	919	880	876

また，下式の条件，すなわち $l_{\downarrow R}$ へのバックジャンプレベルを示す節が学習されない場合には対応する解消リテラルには割当てが含意されないことになる．

$$l_{\downarrow R} \neq \min_{c \in \boldsymbol{L}} l_{\downarrow}(c) \tag{5}$$

しかし，少なくとも 1 つの矛盾は即座に解消したいので，この場合は解消リテラルの一つに対する決定による割当てをバックジャンプ直後に行うことにした．

最後に，解消リテラルが複数の矛盾解消節から選ばれている場合，このリテラルを選ぶ学習節は重要性が高いと考えられるため，そのリテラルを持つ学習節の活性度を高くすることにした．これは，矛盾を複数得ることではじめて実行できる処理であり，単に矛盾を系列として処理することでは得られらない情報を持っている可能性がある．

4. 評価実験

この章では前節のアルゴリズムを用いた提案手法の評価を行う．実験では我々が開発中の SAT ソルバ Mios[楢崎 16] を使用した．これは，基本的に Minisat 1.14[Een03] をモデルに関数型言語 Haskell で実装したソルバである．現実的な問題において Minisat 1.14 に対しておよそ半分の性能を持つが，これはほぼプログラミング言語の差によるものと思われ，監視リテラル，リスタート，VSIDS 変数活性度，など多くの高速 SAT ソルバの高速化手法 [鍋島 10] を共有している．本論文で述べた点以外の変更点はない．

なお，評価尺度として実実行時間ではなく論理制約伝播回数を用いる．これは新規に追加したコードが十分に高速化されていないこと，複数の性能の違う CPU で実行したことによる．SAT ソルバの実行時間の大半は論理制約伝播に費やされているため，この値は実行時間と高い相関を持つ．使用したソルバは非決定性を持たないため，試行はそれぞれ 1 回である．

前章で説明したように提案手法は $p_1 \sim p_4$ の制御パラメータを持つ．以下では p_1 を $1, 2, \cdots 7$ と変え，それぞれで $p_2 \in \{0.1, 0.15, \cdots 0.45\}, p_3 \in \{1, 2, \cdots 8\}, p_4 \in \{1, 2, \cdots 8\}$ の値域（512 点）の中での

- 改善（悪化）が見られたパラメータ設定の個数
- 最もよい結果を得たパラメータ設定での各種統計データ

を示す．パラメータ組 $(0, 0, 0, 0)$ で提案手法の未使用を表す．

4.1 250 変数 3SAT 問題

[Satlib] から入手できる 100 問の 250 変数 3SAT 問題の実行結果の平均値を表 1 に示す．$p_1 = 1.0$ 以外の設定全てで改善が見られたが，改善が見られるのは設定点の 1 割程度であり，適切なパラメータの設定が出来なければむしろ速度の悪化を招くことがわかる．p_1 の変化に対して最良値を与えるパラメータは変動が大きいため，パラメータの選択は難しい課題である．さらに基準として用いた提案手法未実装ソルバでの実験結果は Minisat などでの経験的に選ばれたパラメータを踏襲したものであり，必ずしも真の最良値とは言えないため，問題と相性のよい設定値を確率的に拾っただけという可能性も否定できない．

一方，注目すべきは制御パラメータ p_3 で必ずしも大きな値が選ばれていないという点である．これらは大きな値であるほど学習節をより多く追加する．小さな値がいいということは解析により得た知識であっても効果が薄いと思われるものは積極的に捨てるという方針を支持している．p_4 に関しては値が常に最小値である 1 とならないことから，極性への介入にも効果があることがわかる．ただし，この方針を使わない $p_4 = 0$ と比較して悪くなる場合もあり，その妥当性は十分には確認できていない．

また，これらの値の変更は大きな性能の変化となり，パラメータ安定性が悪いということがわかった．

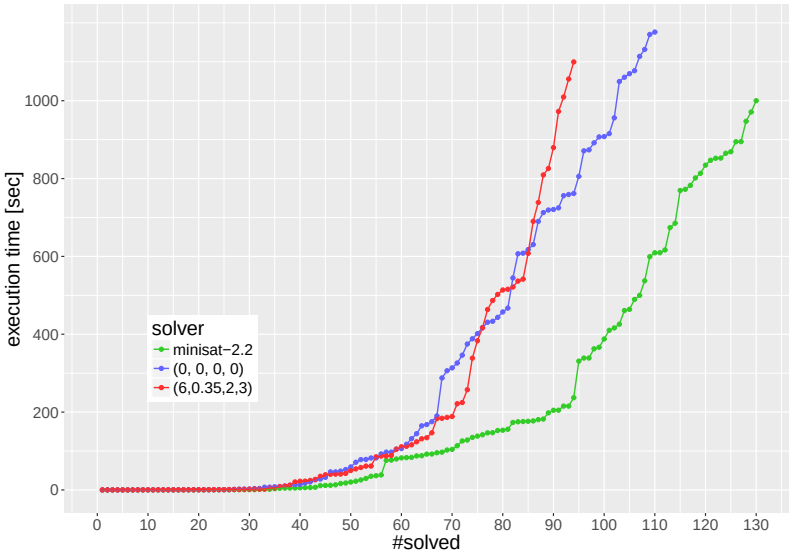

図 3: SAT-RACE 2015 メイントラック（一部）での実行結果

4.2 現実的問題例

次にある現実的な問題に対する実験結果を表 2 に示す．用いた問題は [SAT-Race15] などのベンチマーク問題として用いられているものの一つの `38bits_10.dimacs.cnf` である．これは 384 変数，12700 節 ($n/m = 28.35$), 充足可能，多くの高速ソルバで数秒で解ける問題である．

前節とは最適なパラメータの設定が違うものの，広い範囲で実装前の伝播回数を 1 桁下回ることが出来た．ただし，不適切なパラメータを設定した場合は 1 桁以上の悪化も見られた．制御パラメータの変動，選択される p_3, p_4 など多くの傾向に関して前節と共通の結論を得る．

4.3 SAT-RACE 2015 メイントラック（一部）

最後に SAT-RACE 2015 のメイントラックから Minisat 2.2 が 1000 秒程度で解ける 131 問の部分問題集合を対象として実際の速度を計測した結果を図 3 に示す．パラメータ値は 4.2 節で最もよい結果となった値をそのまま用いた．結果は大きく悪化することになった．現在の解消節分類は静的に与えられる p_3, p_4 のみを用いて実行しているため，様々な変数数や節数の問題から構成されたベンチマークでは改善が見られる問題の方が少ない．問題および求解状況に応じたアルゴリズムの導入が必須である．

5. まとめ

本論文では新たな探索手法である多矛盾バックジャンプ法を提案した．実験による評価ではよい結果を得る場合もあるが，制御パラメータの選択の点で課題があることがわかった．改善が見られるパラメータ域が狭く，問題による差が大きいため，実際的なベンチマークではよい結果が得られていない．

通常の求解過程において学習節は膨大に見つかるものであり，如何に有効でない学習節を選択し削除するかはこれまでも大きな研究テーマの一つであった．これを踏まえると，提案手法の特徴である矛盾を複数得ることによる効果は単に矛盾検出の速度を向上させることではなく，それらの同時解析によって捨てるべき情報と残すべき情報の判断を可能にすることだと思われる．ただし，その判断には十分に問題の求解過程を反映させる必要がある．現在のところ，貪欲伝播部に関しては問題の実行状況に合わせた制御パラメータの設定を試みているが，解消節分類部に関しては十分な検討ができていない．そのため種々の問題が混在するベンチマークではよい結果が出せていないので，今後はこの実験結果を踏まえたより進んだ制御による性能の改善を目指したい．

参考文献

[Audemard09] Gilles Audemard, Laurent Simon: “Predicting Learnt Clauses Quality in Modern SAT Solvers.” IJCAI. vol. 9. pp.399-404, 2009.

[Biere09] Armin Biere et. al: *Handbook of Satisfiability*, IOS Press, 2009.

[Een03] N. Een, N. Sorensson: “An extensible SAT-solver,” in *6th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT2003)*, pp. 502-518, 2003.

[Satlib] `http://www.cs.ubc.ca/~hoos/SATLIB/benchm.html`

[SAT-Race15] `http://baldur.iti.kit.edu/sat-race-2015/downloads/sr15bench.zip`

[Silva99] Marques-Silva, João P., and Karem A. Sakallah: GRASP: A search algorithm for propositional satisfiability, *IEEE Transactions on Computers* , vol. 48, no.5, pp.506-521, 1999.

[鍋島 10] 鍋島英知，栄剛秀: 高速 SAT ソルバーの原理，人工知能学会誌，25 巻 1 号, pp.68–76, 2010-01.

[楢崎 16] 楢崎修二: 関数型言語 Haskell による SAT ソルバーの実装と比較，JSAI2016, 人工知能学会, 1F3-2, 2016-06-06.

[楢崎 17] 楢崎修二: CDCL SAT ソルバーにおけるバックトラック条件の緩和，情報処理学会 2017 年全国大会, 3C-01, 2017-03-18.