3J1-OS7-4

基数制約の概念を持つSATソルバの設計と評価

An Extended SAT Solver for Cardinality Constraints

山根裕二*1 Yuji YAMANE
徐暁雋*1 XiaoJuan XU
上田和紀*2 Kazunori UEDA

*1早稲田大学大学院情報理工学専攻
Department of Computer Science and Engineering, Waseda University

*2早稲田大学理工学術院情報理工学科
Department of Computer Science and Engineering, Waseda University

Cardinality constraints generated from real-world problems are often a bottleneck of SAT solving. This paper proposes an extended CNF that can express cardinality constraints and allows high-performance unit propagation. We have implemented it on MiniSAT 09z, a variant of MiniSat 2.0 which is one of the most popular SAT solvers, by improving deduction and propagation to work with the extended CNF. To evaluate the performance improvement over the original MiniSAT, we made an experiment on this extented solver with the ITC2007 benchmark set that contains cardinality constraints problems.

1. はじめに

命題論理の充足可能性問題 (Boolean Satisfiability Problem: SAT) は，代表的な NP 完全問題である．近年高速化が著しい SAT ソルバは，ソフトウェア検証やハードウェア検証のエンジンとしての利用が期待されている．実問題は基数制約 (Cardinality Constraints) が多く含まれる場合があるが，そのような問題を SAT に変換して解かせようとすると，真偽値のみを扱う SAT ソルバでは，制約の表現に多くの節を必要とし，実行中にメモリ不足に陥ることがある．本研究では，基数制約を含む実問題に対してよりスケーラブルかつ高速な求解を目的に，SAT ソルバに基数制約を表す特殊な節を加えることを提案する．SAT ソルバの既存のアルゴリズムを特殊節に対し適用できるように拡張，実装し，基数制約を含む問題として代表的な時間割作成問題を拡張前と後両方の SAT ソルバにそれぞれ解かせて比較することで，評価を行った．

2. SAT

SAT は，ある命題論理式が充足可能 (SAT) になるような変数割当を求めるか，どのような割当をしても充足不能 (UNSAT) であることを証明する問題である．SAT は乗法標準形 (Conjunctive Normal Form: CNF) を入力としている．CNF は節の集合であり，節はリテラルの集合であり，リテラルは 2 値の変数の正負の出現を表す．

2.1 CDCL ソルバ

昨今の SAT 競技会においては，大多数のソルバが DPLL 手続きに学習機構を組み込んだ CDCL (Conflict-Driven Clause Learning) アルゴリズムを採用している．具体的には，現在の真偽値割当において単位節となる節に対し，残っている 1 つのリテラルに真を割り当てる．単位節が無くなるか矛盾が生じるまでこの作業を繰り返す (単位伝播)．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，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．単位節が発生するまで割当を繰り返す．矛盾が生じた場合は矛盾を引き起こした原因を解析する．解析の結果，矛盾を引き起こした割当の集合を発見すると，その集合の否定を学習節として問題に追加し，更に学習節を単位節とするような探索レベルまでバックトラックを行う．学習節の存在によって矛盾を起こした割当を回避できるので，探索空間の剪定が可能になる．CDCL ソルバの擬似コードを [10] より引用し，図 1 に載せる．dl は現在の決定レベルを表し，propagate 関数は単位伝播を行って変数割当てを返す (3.2 節)，analyze 関数は矛盾の原因を解析して学習節およびどのレベルまでバックトラックすべきかを C と bl にそれぞれ返す．lv 関数は変数値の決定レベルを返す．

```
CDCL(ψ, ν)
ψ: a CNF formula, ν: a variable assignment
begin
  dl := 0
  loop
    ν = propagate(ψ, ν)
    if ν(ψ) = 1 then return 1
    if ν(ψ) = 0 then
      if dl = 0 then return 0
      (C, bl) := analyze(ψ, ν)
      ψ := ψ ∪ {C}
      ν := {(x_i, v_i) ∈ ν | lv(x_i) ≤ bl }
    dl := bl
    else
      dl := dl + 1
      choose an unassigned variable x and its
      value v
      ν := ν ∪ {(x, v)}
end
```

図 1: 一般的な CDCL アルゴリズム

連絡先: 山根裕二，早稲田大学大学院基幹理工学研究科情報理工学専攻，〒169-8555 東京都新宿区大久保 3-4-1，03-5286-3340，yuji(at)ueda.info.waseda.ac.jp

2.2 SAT における基数制約

基数制約とは，0 か 1 を取る 2 値の整数変数の有限集合 $x_1, x_2, \ldots$ と正の定数 k が存在するとし，

$$\sum_{i=1}^{n} x_i \leq k \quad (\text{または} \sum_{i=1}^{n} l_i \geq k)$$

と記述される制約である [8]．一般に時間割問題や配置問題などの実問題で整数を取り扱う場面は多く，実問題と基数制約には密接な関係があると言える．SAT では真偽値のみを扱うので，単純な変換方法を用いると節の数は指数的に増加する．通常の CNF を用いて基数制約をコンパクトに表現する手法なども提唱されているが [6]，本研究では SAT ソルバに対し基数制約を 1 つの節として表現できるように，CNF の記法を拡張した．

3. SAT の拡張

SAT の拡張とは，CNF の記法の拡張とそれを適用させるための SAT ソルバの実装の変更を意味する．まずは，既存の DIMACS 形式の CNF に基数制約を表す特殊節を追加することを考える．基数制約を表現する特殊節とは，その節のリテラルの k 個以上が真であることを意味する at-least-k 節とその節のリテラルの k 個以下が真であることを意味する at-most-k 節を指す．従来の CNF における節は，その節に含まれるリテラルのうち 1 個以上が真となることを表しているので，at-least-one 節に分類され，特殊節に対しこれを通常節と呼ぶこととする．

3.1 特殊節の記法

C をリテラルの集合とする at-least-k 節を $C_{\geq k}$ と表記することとする．通常節は $C_{\geq 1}$ と表記できる．そして at-most-k 節は $C_{\leq k}$ と表記することにする．ここで，特殊節の性質を以下に記す．

$$C_{\geq k} = \overline{C}_{\leq n-k} \tag{1}$$

$$\{A \cup (\bigcup_{i=1}^{k} l_i)\}_{\geq x} \otimes \{B \cup (\bigcup_{i=1}^{k} \neg l_i)\}_{\geq y} = \{A \cup B\}_{\geq x+y-k} \quad (x, y, k \in N \quad k \leq min(x, y)) \tag{2}$$

式 (1) の性質は，n 個のリテラルを持つ at-least-k 節はそのリテラルを全て正負について反転させた at-most-$(n-k)$ 節と等しいことを意味する．つまり，at-least-k 節または at-most-k 節のどちらか一つを採用することで，基数制約をコンパクトに表現することが出来る．本研究では at-least-k 節だけを採用し，以降は特殊節の性質を述べる際は at-least-k 節の場合だけを考えることとする．式 (2) は以降の章で述べる単位伝播や矛盾解析において重要な性質を示している．$\otimes$ は 2 つの節の resolution を意味している．あるリテラルの有限集合 A を含む at-least-x 節とあるリテラルの有限集合 B を含む at-least-y 節がそれぞれ互いに正負の反転したリテラルを k 個持つ場合，resolution すると A と B の和集合の全要素をリテラルとして含む at-least-$(x+y-k)$ 節となる．$\{A \cup (\bigcup_{i=1}^{k} l_i)\}_{\geq x}$ と $\{B \cup (\bigcup_{i=1}^{k} \neg l_i)\}_{\geq y}$ を両方真にすることを考えたとき，$\{A \cup B \cup (\bigcup_{i=1}^{k} l_i) \cup (\bigcup_{i=1}^{k} \neg l_i)\}_{\geq x+y}$ を満たす必要があるが，この式は k 個の排中律を含むため式 (2) のように変形できる．

3.2 特殊節における単位伝播

特殊節を含む CNF の単位伝播は，通常節の単位伝播に引き続いて特殊節の単位伝播を行うことで実現している．それらのコードを，[10] に基づく記法で図 2 と図 3 に載せる．関数 propagate は入力として CNF 形式の論理式 ψ と値の割り当ての組み合わせ ν を受け取り，ν に新たな値の割り当ての組み合わせを追加して返す．$\nu(\psi)$ は 1 のときは充足可能，0 のときは充足不能を意味する．未定リテラルを 1 つだけ含む節を単位節として，リテラルを真にするような変数割当を繰り返す．関数 propagate_LC は関数 propagate と振る舞いは変わらないが，at-least-k 節が未定リテラルを k 個含む場合，それを単位節と見なし k 個全てのリテラルを正にするような変数割当を行う点で異なる．変数伝播の流れを 3.1 節の式 (2) に則って説明すると，A (または B) が空集合であり k が x (または y) と等しい場合となる．従来は SAT ソルバの処理の 80%を変数伝播の処理が占めるとされており，関数 propagate_LC を導入し効率的な変数伝播を行うことで高速化を図る．SAT ソルバに拡張を行う事で拡張前に比べ通常の CNF を解く速度の低下が考えられるが，今回の実験においては速度の低下は平均で 5% に満たなかった (4.3 節)．

```
propagate(ψ, ν)
ψ: a CNF formula, ν: a variable assignment
begin
  while ∃ C_≥1 ∈ ψ (C_≥1 is a unit clause) and
  ν(ψ) ≠ 0 do
    l := the unassigned literal in C_≥1
    ν := ν ∪ {(var(l), sign(l))}
  if ν(ψ) = 1 then
    ν = propagate_LC(ψ, ν)
  return ν
end
```

図 2: 通常節における propagation のアルゴリズム

```
propagate_LC(ψ, ν)
begin
  while ∃ C_≥k ∈ ψ (C_≥k is a k-ary clause) and
  ν(ψ) ≠ 0 do
    while ∃ l ∈ C_≥k (l is an unassigned literal)
      ν := ν ∪ {(var(l), sign(l))}
  return ν
end
```

図 3: at-least-k 節における propagation のアルゴリズム

3.3 特殊節を含む場合の矛盾解析

現在の SAT ソルバの矛盾解析では，学習節は resolution プロセスによって生成される．矛盾を起こした節とその単位伝播の原因となった節 (これを，reason clause と呼ぶことにする) で resolution を行い，新たに生成した節とその節の reason clause とで resolution を取るという処理を繰り返し，新たに生成した節に含まれるリテラルのうち 1 つを除く全てのリテラルが現在の決定レベルよりも浅いとき，その節を C とし，C を簡単化したものが学習節となる．特殊節を含む場合も同様の resolution プロセスを取っており，その resolution 規則は 3.1 節の式 (2) に記している．特殊節を含む場合，新たに生成

した節が at-least-t 節でその節に含まれるリテラルのうち t 個を除くすべてのリテラルが現在の決定レベルよりも浅いとき，その節を $C_{\geq t}$ とし，$C_{\geq t}$ を簡単化したものが学習節となる．

3.4 関連研究

基数制約を扱う先行研究としては，擬似ブール制約 (Pseudo-Boolean Constraints) が挙げられる．擬似ブール制約は変数がブール値を取る制約充足問題であり，$\sum_{i=1}^{N} a_i x_i \geq b$ の形の線形制約を主な対象とする．a_i と b は整数，x_i はリテラルであり，真のとき 1，偽のとき 0 と解釈する．$a_i = 1, i \in (1, 2, ..., N)$ のとき，基数制約とみなせることからも明らかなように，擬似ブールソルバでは基数制約をそのまま記述可能である．擬似ブールソルバとしては MiniSat+ [2] などが代表的であるが，本稿での実験で利用したベンチマーク問題を解かせたところ，いずれの問題においても求解に膨大な時間を要し，通常の SAT ソルバと比較すると速度に大きな差があった．

特殊な節の導入に関しての先行研究では，基本対称関数に基づく節 (ES 節と呼ばれる) の導入による CNF の拡張が挙げられる [9]．ES 節とは「n 個の変数のうち，ちょうど k 個が真」という制約を意味する節を表す．この研究に対する本研究の新規性としては，at-least-k 節と at-most-k 節をそれぞれ分割しているため ES 節よりも柔軟に制約の記述が可能な点や，特殊節に対して監視リテラルを設けている点，既存の SAT ソルバである MiniSat に対して拡張，実装している点などが挙げられる．

特殊な節の導入に関する先行研究としては，他にも SAT-Race 2010 の Main Track 部門で優勝した CryptoMiniSat [7] が挙げられる．CryptoMiniSat は排他的論理和を表す特殊節を導入することで暗号問題などにおいて高速な処理を可能にしている．

4. 実装と評価実験

4.1 実装

3.1 節の式 (1) の性質より，at-least-k 節があれば at-most-k 節を表現できる．そこで，at-least-k 節のみを追加することとする．at-least-k 節の実装にあたり，節を保持する構造体にその節を真にするために必要な真リテラルの数を格納するメンバ変数を追加した．また，at-least-k 節では $k+1$ 個のリテラルを監視することとしている．

4.2 実験概要

基数制約がボトルネックになるような問題を対象として，SAT ソルバと特殊節に対応する拡張 SAT ソルバの比較実験を行う．

4.2.1 ITC 2007

基数制約を多く含む代表的な問題として，時間割作成問題の競技会である International Timetabling Conpetiton 2007 (ITC 2007) [3] の大学の講義の時間割作成問題である Post Enrolment Course Timetabling 部門で提供されるベンチマーク 24 題を選んだ．数百人の生徒が特定の教室で特定の講義を履修する状況を想定し，数百もの講義に対して 45 のタイムスロット (5 日制で 9 時限) で定められる日程と教室を適切に割り当てることが求められている．

4.2.2 最適解の探索

ベンチマーク問題には必ず満たす必要のある Hard Constraints (H1 ～ H5) と満たしていないとペナルティとなる Soft Constraints (S1 ～ S3) が存在する．以降に，Hard Constraints と Soft Constraints を記す．

- H1: どの学生も同時刻に 2 つ以上の授業を履修することはない．
- H2: 教室は常に履修人数以上のサイズであり，授業が教室に求める特徴を全て満たしている．
- H3: 同時刻に同じ教室で 2 つ以上の授業は行われない．
- H4: 各授業が実施可能なタイムスロットに割当てられている．
- H5: 各授業が指定された正しい順序で割当てられている．
- S1: 生徒は，1 日の最後の時限に授業を取らない．
- S2: 生徒は，3 つ以上連続して授業を取らない．
- S3: 生徒は，1 日に 1 つの授業だけを取ることはない．

上述の制約を完全に満たすような最適解が必ず 1 つ存在することが明記されており，本実験では最適解を求めることを目標にしている．ただし，ベンチマーク問題はもともと大規模な問題なので最適解を求めることは難しく，独自に生徒の数を変数として値を変更することで問題の縮小化を行っている．

4.2.3 特殊節の導入による節数の削減効果

通常節のみを用いてベンチマーク問題の制約を単純な変換方法で CNF に変換した場合の節数のオーダーと，特殊節を用いた場合の節数のオーダーを表 1 に，記す．表中の変数 e は授業の数，s は生徒の数，k は教室の数，$|\overline{C_m}|$ は生徒の授業の履修数の平均値を表す．制約 H1 と S2 と S3 が基数制約であり，H1 は at-most-one 節，S2 は at-most-two 節，S3 は at-least-two 節に対応している．特に，S2 と S3 は従来の CNF 形式に変換したときの節数の 9 割前後を占めており，この 2 制約が大きなボトルネックと言える．生徒の授業の履修数の平均値は各問題では 10 前後であり，特殊節を導入することで実質的に節数を 1% から 10% 前後にまで減らすことが可能である．

表 1: 特殊節の導入による節数の削減効果

制約	通常節のみ	通常節+特殊節
H1	$O(e^2 \cdot s)$	$O(s)$
H2	$O(e)$	$O(e)$
H3	$O(e^2 \cdot k)$	$O(e^2 \cdot k)$
H4	$O(e)$	$O(e)$
H5	$O(1)$	$O(1)$
S1	$O(e)$	$O(e)$
S2	$O(s \cdot \lvert\overline{C_m}\rvert^3)$	$O(s \cdot \lvert\overline{C_m}\rvert)$
S3	$O(s \cdot \lvert\overline{C_m}\rvert^2)$	$O(s \cdot \lvert\overline{C_m}\rvert)$

4.2.4 実験環境

ITC 2007 のインスタンスを CNF と拡張 CNF に変換して，前者を従来の SAT ソルバに，後者を提案手法により拡張させた SAT ソルバにそれぞれ解かせて結果を比較した．本稿執筆時点では，拡張 SAT ソルバは at-least-two 節のみに対応している．実験環境は表 2 に記している．

表中からもわかるように，SAT ソルバには MiniSAT 09z [4] を用いている．MiniSAT 09z は標準的な SAT ソルバである MiniSat2.0 を元にリスタート戦略を変更しており，2009 年の MiniSat Hack Track 部門で優勝した高速なソルバである．

表 2: 実験環境

OS	CentOS 5.3
CPU	Intel Xeon(R) X5650 2.67GHz, 6 Core × 2
メモリ容量	48GB
gcc version	4.1.2
SAT ソルバ	MiniSAT 09z

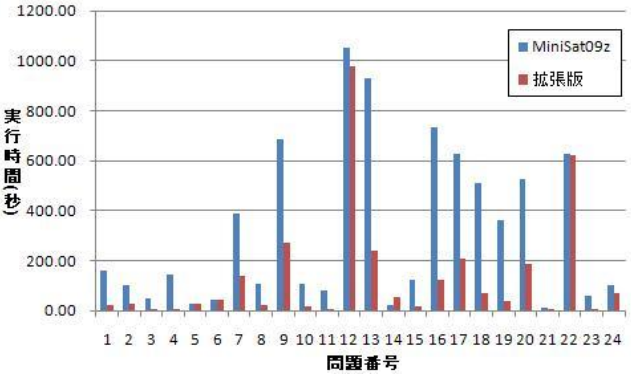

図 4: MiniSAT 09z と拡張版の比較

4.3 実験結果

実験結果を図 4 に載せ，合計時間を表 3 に載せる．24 問のいずれも，ITC2007 で提供されるベンチマーク問題の生徒数を減らしている．生徒数の縮小は，実行時間が 10 秒以上でかつ 1200 秒を超えない間に収まる程度に調整している．2 回または 3 回の実験のうち、もっとも良い数値を採用しているが，各結果にほとんど差は無い．表 3 には，通常の CNF を拡張版 MiniSAT 09z で解いた場合の合計時間も比較データとして載せている．節の構造体に変数を追加した分速度が若干低下しているものの，高々 5% 程度に留まっていた．各問題の実行時間を比較すると，拡張版 MiniSAT 09z では算術平均で 6.2 倍，幾何平均で 3.8 倍の速度向上が見られた．また，特定の問題 (図 4 における問題 15) を固定して，生徒数を調整させつつ MiniSAT 09z と拡張版を比較した結果が図 5 である．こちらの実験では制限時間を 3600 秒としている．充足可能な問題であり問題のサイズに応じて実行時間が単調増加しない場合もあるが，全体の傾向としては問題のサイズを大きくすると実行時間はほぼ指数関数的に増加することが見て取れる．拡張版は拡張前に比べて多少大きなサイズでも処理可能であるが,，問題のサイズの増加に伴い実行時間も大幅に増加する点は変わらない．

5. まとめと今後の課題

基数制約を表す特殊節を含む拡張 CNF を解く SAT ソルバを設計し，at-least-two 節に対応する段階まで実装を行った．基数制約を多く含む時間割作成問題をベンチマークとして拡張版 MiniSAT 09z と従来の MiniSAT 09z とを比較したところ，算術平均で 6.2 倍，幾何平均で 3.8 倍の速度向上が得られた．基数制約は SAT においては節数の増大を招くことが想定され，それを 1 つの節の構造体にまとめることで実行速度が向上していた．また，最高で 23.9 倍もの速度向上が得られた．問題ごとに速度の差が出るのは at-least-two 節の部分が全体の節数に対しどれだけの割合を占めるかに依存するものと推測しているが，さらなる調査が必要である．

表 3: MiniSAT 09z と拡張版の比較 (合計時間)

通常の CNF		拡張 CNF
拡張版 09z	MiniSAT 09z	拡張版 09z
7985.8 秒	7604.9 秒	3212.0 秒

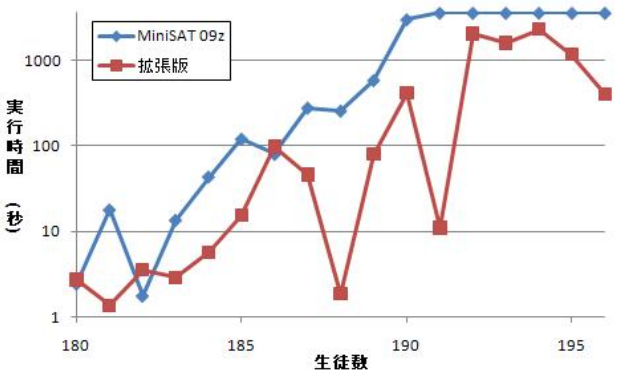

図 5: MiniSAT 09z と拡張版の比較 (特定の問題)

今回の実装では at-least-two 節にのみ対応しているが，今回の実験結果から一般的な at-least-k 節に対応することで更なる高速な実行が可能になると考えられる．さらに，特殊節の導入による速度向上を確認できたので，今後は MiniSat 2.2 などの高速な SAT ソルバを拡張し評価実験を行っていく．また，より大規模な実問題の求解を目指してクラスタ環境向けの並列 SAT ソルバである c-sat [5] のアイデアを元に，特殊節を扱う並列 SAT ソルバの実装を今後の課題とする．

参考文献

[1] Eén, N. and Sörensson, N.: An Extensible SAT-solver, in *Proc. SAT 2003*, LNCS 2919, Springer, pp. 502–518, 2004.

[2] Eén, N. and Sörensson, N.: Translating Pseudo-Boolean Constraints into SAT, *Journal on Satisfiability, Boolean Modeling and Computation*, Vol. 2, pp. 1–26, 2006.

[3] International Timetabling Competition 2007, 2007. `http://www.cs.qub.ac.uk/itc2007/`

[4] Iser, M.: MiniSAT 09z for SAT-Competition 2009, in SAT 2009 competitive events booklet, pp. 29–30, 2009.

[5] Ohmura, K. and Ueda, K.: c-sat: A Parallel SAT Solver for Clusters, in *Proc. SAT 2009*, LNCS 5584, Springer, pp. 524–537, 2009.

[6] Sinz, C.: Towards an Optimal CNF Encoding of Boolean Cardinality Constraints, in *Proc. CP 2005*, pp. 827–831, 2005.

[7] Soos, M.: CryptoMiniSat 2.5.0, 2007. `http://baldur.iti.uka.de/sat-race-2010/descriptions/solver_13.pdf`

[8] 井上克巳, 田村直之: SAT ソルバーの基礎, 人工知能学会誌, Vol. 25, No. 1 (2010), pp. 57–67, 2010.

[9] 馬野洋平, 酒井正彦, 西田直樹, 坂部俊樹, 草刈圭一朗: 基本対称関数を付加した CNF 論理式の充足可能性判定, IEICE Transactions on Information and Systems, Vol. J93–D, No. 1, pp. 1–9, 2010.

[10] 鍋島英知, 宋剛秀: 高速 SAT ソルバーの原理, 人工知能学会誌, Vol. 25, No. 1(2010), pp. 68–76, 2010.