コンピュータ科学科
情報システム系
卒業論文

SAT符号化を用いた命題論理式の極小・極大モデル計算の実装と評価

2026年2月

102230349　前田 悠士朗

概要

本論文では，SAT 符号化を用いた命題論理式の極小・極大モデルの効率的な計算方法について述べる．極小モデルとは，CNF で与えられる命題論理式のモデル集合に包含関係を入れて定義され，真となる変数をこれ以上減らせないモデルである．また，極大モデルとは，真となる変数をこれ以上増やせないモデルである．極小モデルや極大モデルを計算することは整数ナップサック問題の圧縮解計算や最適化問題の多目的パレートフロント求解において有用である．従来の反復起動法はモデル候補ごとに SAT ソルバー起動が増え計算コストが大きい．本研究では，負リテラルに着目し他リテラルで節が満たされるなら当該変数を真にする制約を追加することで，SAT ソルバー 1 回の起動で極大モデルを計算できる極大変換を提案した．また，既存の SAT ソルバー 1 回の起動による極小モデル計算と提案する極大モデル計算を実装し，$3 \times n$ のグリッドグラフを用いて評価実験を行った．その結果，SAT ソルバーの起動回数が大幅に削減され，反復起動法よりも多くの問題を制限時間内に解けることを確認した．

目 次

図 目 次

表 目 次

第1章　はじめに

モデルとは，命題変数からなる論理式に対して，各変数に真偽値を割り当てることにより，その論理式を真にする割り当てのことである．

命題論理式の充足可能性判定問題（SAT 問題）とは，与えられた命題論理式に対して，このようなモデルが存在するかどうかを判定する問題である．近年は SAT ソルバーの高性能化により，大規模かつ複雑な論理式に対しても実用的な時間でモデルを得られるようになっており，様々な分野で推論基盤として用いられている [1, 2, 3, 7, 8]．一方で，実際の応用においては，単にモデルが存在するかどうかだけでなく，得られるモデルの中からどのような代表的解を抽出するかが重要となる．

極小・極大モデルとは，命題論理式のモデル集合に包含関係を導入して定義される，真の変数集合をこれ以上減らせないモデル，および真の変数集合をこれ以上増やせないモデルである．極小・極大モデルは冗長性のない代表的なモデルと捉えることもでき，論理プログラミングの非単調推論や circumscription に基づく常識推論の意味論的基盤を成すとともに，グラフ理論における極小支配集合や極大独立集合，多目的最適化におけるパレートフロントの計算など，幅広い応用と対応している [5, 6]．これらの応用において，極小・極大モデルを高速に計算することは重要な研究課題である．

既存の計算方法 [4] として，SAT ソルバーを複数回起動して徐々にモデルを包含関係において大きく・小さくする方法で極小・極大モデルを計算する方法が提案された．しかし，この方法ではモデル候補ごとに SAT ソルバーの起動が増加し，さらに極小性・極大性を保証するために充足不能性の判定を行う必要があるため，計算コストが大きくなるという課題がある．

そこで，SAT 符号化を用いて SAT ソルバーを 1 回起動するのみで極小モデルおよび極大モデルを計算する方法に注目する．この方法に関する足立の先行研究では，与えられた連言標準形の命題論理式 (CNF 式) Ψ に適切な変換を加えた $f(\Psi)$ を得ることで，Ψ の極小モデルが $f(\Psi)$ のモ

デルと一致することが示された．しかし，このような SAT 符号化に基づく手法については，極小モデルに関するものが中心であり，極大モデルに対する同様の手法や，それらの性能評価は十分に行われていなかった．

本論文では，SAT 符号化を用いて SAT ソルバーを 1 回起動することで極大モデルを計算するための CNF 変換方法を提案する．また，SAT ソルバーの 1 回起動および複数回起動に基づく極小モデル計算・極大モデル計算手法の実装を行い，グラフ問題に基づくベンチマークを用いた実行実験による従来手法との性能比較を行った．SAT ソルバー 1 回起動による極小モデルの計算方法では，SAT 問題に符号化を施した CNF の正リテラルに注目し，ある節中の他の変数で節が満たされているとき，その変数を偽にするという制約を追加することにより，極小モデル計算を実現した．また，SAT ソルバー 1 回起動による極大モデルの計算方法では，SAT 問題に符号化を施した CNF の負リテラルに注目し，ある節中の他の変数で節が満たされているとき，その変数を真にするという制約を追加することにより，極大モデル計算を実現した．

本論文の主な貢献と結果は以下の通りである．

- 極大モデルに対しても SAT 符号化を用いて SAT ソルバーを 1 回だけ起動することで計算する方法を提案した．足立 [10] の極小モデルの変換では各正リテラルに対して制約を追加するのに対し，本手法では各負リテラルに対して制約の追加を行う．これにより先行研究と併せて極小・極大モデルの両方を計算するための基盤を構築した．

- 符号化を用いて極小・極大モデルを列挙する手法を SAT ソルバーと Rust 言語を用いて実装した．このプログラムは CNF 式を入力とし，提案する符号化を適用した上で SAT ソルバーを呼び出すことにより極小・極大モデルを列挙する．

- 提案手法の有効性を検証するために，極小支配集合問題 (MDS) と極大独立集合問題 (MIS) の計 50 問のベンチマーク問題を用いて評価を行った．既存手法と提案手法とで全ての極小・極大モデルを列挙する CPU 時間を比較した結果，提案手法は既存手法よりも多くの問題を高速に解くことに成功し，符号化を用いた方法の有効性を確認した．

提案手法は，SAT ソルバーの単一呼び出しで極小・極大モデルを計算可能にするものであり，組合せ問題などにおける解の計算技術に寄与するものである．

第2章　命題論理式の極小・極大モデル

2.1　SAT 問題の定義

本節では，論文 [9] をもとに SAT 問題を定義する.
命題論理式の充足可能性判定問題 (SAT 問題; Boolean Satisfiability Testing Problem) とは，与えられた命題論理式を充足するような命題変数への値の割り当てが存在するかどうかを判定する問題である.

定義 1 (命題変数) **命題変数**は，1 または 0 の値をとる変数であり，それぞれ**真 (true)**，**偽 (false)** を表す.

定義 2 (命題論理式) 命題変数の集合を V とする．**命題論理式**の集合 $\mathrm{Form}(V)$ を次の生成規則により再帰的に定義する.

- （原子式）任意の $x \in V$ は命題論理式である（すなわち $x \in \mathrm{Form}(V)$）.
- （否定）$\psi \in \mathrm{Form}(V)$ ならば，$\neg\psi \in \mathrm{Form}(V)$.
- （二項結合）$\psi_1, \psi_2 \in \mathrm{Form}(V)$ ならば，$(\psi_1 \wedge \psi_2)$，$(\psi_1 \vee \psi_2)$，$(\psi_1 \to \psi_2)$ は命題論理式である.

ここで用いる論理結合子は表 2.1 に示す 4 種類である.

定義 3 (充足) V に対する真偽値割り当て（解釈）を $I : V \to \{1, 0\}$ とする．任意の命題論理式 $\psi \in \mathrm{Form}(V)$ に対し，I が ψ を充足すること（$I \models \psi$）を次のように再帰的に定義する．また，$I \models \psi$ が成り立たないことを $I \not\models \psi$ と表す.

- （原子式）$I \models x \Leftrightarrow I(x) = 1 \quad (x \in V)$

表 2.1: 論理結合子

記号	意味	英語
$\wedge$	かつ，連言	and, conjunction
$\vee$	または，選言	or, disjunction
$\rightarrow$	ならば，含意	implies, implication
$\neg$	でない，否定	not, negation

- （否定）$I \models \neg\psi \Leftrightarrow I \not\models \psi$
- （連言）$I \models (\psi_1 \wedge \psi_2) \Leftrightarrow (I \models \psi_1$ かつ $I \models \psi_2)$
- （選言）$I \models (\psi_1 \vee \psi_2) \Leftrightarrow (I \models \psi_1$ または $I \models \psi_2)$
- （含意）$I \models (\psi_1 \rightarrow \psi_2) \Leftrightarrow (I \not\models \psi_1$ または $I \models \psi_2)$

定義 4 (モデル) 割り当て I が命題論理式 ψ を充足するとき，I は ψ の**モデル (model)** であるという．

定義 5 (充足可能) 命題論理式 ψ にモデルが存在するとき，命題論理式 ψ は**充足可能 (satisfiable)** であるという．

定義 6 (充足不能) 命題論理式 ψ にモデルが存在しないとき，命題論理式 ψ は**充足不能 (unsatisfiable)** であるという．

例 1 命題変数の集合を $V = \{x_1, x_2\}$，命題論理式 ψ が以下のように与えられているとする．

$$\psi \ = \ (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$$

また，割り当て I_1, I_2 が以下のように与えられているとする．

$$I_1(x_1) = 1, I_1(x_2) = 0$$
$$I_2(x_1) = 0, I_2(x_2) = 1$$

このとき，$I_1 \models (x_1 \vee x_2)$ は真であるが，$I_1 \models (\neg x_1 \vee x_2)$ は偽であるので，$I_1 \not\models \psi$.
一方で，$I_2 \models (x_1 \vee x_2)$，$I_2 \models (\neg x_1 \vee x_2)$ は共に真であるので，$I_2 \models \psi$ より，I_2 は ψ のモデルである．すなわち，ψ は充足可能である．

2.2 CNF の定義

SAT 問題においては，以下で定義する**連言標準形 (Conjunctive Normal Form;CNF)** を入力とすることが一般的であり，本論文でも SAT 問題は CNF で与えられるものとする．

定義 7 (リテラル) **リテラル (literal)** とは，命題変数 x またはその否定 $\neg x$ であり，前者を正リテラル，後者を負リテラルと呼ぶ．

定義 8 (節) **節 (clause)** とは，リテラルの選言 ($\vee$；OR)，すなわち，リテラルを論理和 $\vee$ で結合した論理式である．

定義 9 (CNF) **CNF(Conjunctive Normal Form; 連言標準形)** とは，節の連言（$\wedge$；AND），すなわち，節を論理積 $\wedge$ で結合した論理式である．

例 2 CNFψ が以下のように与えられているとする：

$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_3) \wedge (x_2 \vee x_3)$$

このとき，割り当て $(x_1, x_2, x_3) = (1, 0, 1)$ は ψ を充足するため，ψ は充足可能である（SAT）．他の ψ のモデルとして，$(1, 1, 0)$ と $(1, 1, 1)$ が存在する．
一方，CNF$\psi' = \psi \wedge (\neg x_1)$ は充足不能となる（UNSAT). なぜなら，ψ の任意のモデル I において，$I(x_1) = 1$ であり，$I \not\models \neg x_1$ であるため，$I \not\models \psi'$ となるからである．

2.3 極大モデル，極小モデルの定義

本節では，論文 [11] をもとに極大モデル，極小モデルを定義する．
CNF のモデルは，そのモデルで真と解釈される変数の集合で表現することができる．例えば，x_1 を真，x_2 を偽，x_3 を真，と解釈するモデルは変数集合 $\{x_1, x_3\}$ で表現できる．このように表すことにより，特定のモデル間に包含関係が導入され，モデル間の大小関係を自然に導入できる. 例えば，モデル $\{x_1, x_3\}$ はモデル $\{x_1, x_2, x_3\}$ より包含関係上で小さい. 繰り返しになるが，モデル間にこのような包含関係を導入したとき，極大モデル，極小モデルを以下のようにも表すことができる．また，本論文では，理解のしやすさのためモデル M を次のように定義する．

定義 10 (モデル M) 命題論理式 ψ のモデル $I : V \to \{0, 1\}$ に対し，モデル M を以下のように定義する．

$$M \ = \ \{v \mid I(v) = 1\}$$

定義 11 M_1, M_2 を変数集合とする．このとき，M_1 が M_2 より大きいとは， M_2 が M_1 の真部分集合であることをいう．

定義 12 (極大モデル) ψ を命題論理式， M を ψ のモデルとする． このとき， M が ψ の**極大モデル**である， とは，M より大きい ψ のモデルが存在しないことをいう．

定義 13 M_1, M_2 を変数集合とする．このとき，M_1 が M_2 より小さいとは， M_1 が M_2 の真部分集合であることをいう．

定義 14 (極小モデル) ψ を命題論理式， M を ψ のモデルとする． このとき， M が ψ の**極小モデル**である， とは，M より小さい ψ のモデルが存在しないことをいう．

例 3 2.2 節の例 2 で挙げた CNFψ の場合，極小モデル，極大モデルは表 2.2 のようになる．今回の例のモデルは極小，極大モデルのみであったが，

表 2.2: ψ のモデルと極小・極大モデル

モデル (x_1, x_2, x_3)	真になる変数集合	区分
$(1, 0, 1)$	$\{x_1, x_3\}$	極小モデル
$(1, 1, 0)$	$\{x_1, x_2\}$	極小モデル
$(1, 1, 1)$	$\{x_1, x_2, x_3\}$	極大モデル

一般的な問題では極小，極大モデルのどちらでもないモデルが存在する．

第3章　SATソルバー複数回起動による極小・極大モデル計算の既存研究

本章では，論文 [4] で提案された，SAT ソルバーを複数回起動することで SAT 問題の極小モデルを求める方法について説明する．ここで SAT 問題を符号化した P が与えられたとき，極小モデルは以下の手順で求めることができる．

1. P に SAT ソルバーを起動し，モデル M を求める
2. 1. で求めたモデル M のうち，真として含まれるリテラルを $x_1, ..., x_m$ とし，偽として含まれるリテラルを $y_1, ..., y_n$ とし，次の F1,F2 を作る

$$\begin{aligned} \text{F1} = \ & \neg(x_1 \wedge ... \wedge x_m) \\ \text{F2} = \ & \neg y_1 \wedge ... \wedge \neg y_n \end{aligned}$$

3. $P := P \wedge F1 \wedge F2$ として，SAT ソルバーを起動する
   - UNSAT ならば，モデル M が極小モデル
   - SAT ならば，1. に戻る

アルゴリズムで記述すると Algorithm1 のようになる．

例 4 2.2 節の例 2 で与えられた式を P として，先ほどのアルゴリズムに沿って極小モデルを求めてみる．
（1 回目のループ）

1. P に SAT ソルバーを起動し，$(x_1, x_2, x_3) = (1, 1, 1)$ がモデル M として求められる
2. 1. で求めたモデル M から次の F1，F2 を作る

Algorithm 1 SAT ソルバー複数回起動による極小モデル生成

Input: SAT 問題の符号化 P
Output: P の極小モデル M（存在しなければ UNSAT）

while SOLVE(P) が SAT を返し，モデル M を得る **do**
  $X \leftarrow \{x \mid x$ は M において真となるリテラル$\}$
  $Y \leftarrow \{y \mid y$ は M において偽となるリテラル$\}$
  ▷ $X = \{x_1, \ldots, x_m\}$, $Y = \{y_1, \ldots, y_n\}$ とする
  $F1 \leftarrow \neg(x_1 \wedge \cdots \wedge x_m)$　▷ $\equiv (\neg x_1 \vee \cdots \vee \neg x_m)$
  $F2 \leftarrow (\neg y_1) \wedge \cdots \wedge (\neg y_n)$
  $P' \leftarrow P \wedge F1 \wedge F2$
  if SOLVE(P') が UNSAT **then**
    return M
  else
    $P \leftarrow P'$
  end if
end while
return UNSAT

$$\begin{aligned} \mathrm{F1} &= \neg(x_1 \wedge x_2 \wedge x_3) \\ \mathrm{F2} &= \top \end{aligned}$$

3. $P \coloneqq P \ \wedge \ \neg(x_1 \wedge x_2 \wedge x_3) \ \wedge \ \top$ として，SAT ソルバーを起動すると，SAT であるので，1. に戻る.

(2 回目のループ)

1. $P \coloneqq P \ \wedge \ \neg(x_1 \wedge x_2 \wedge x_3) \ \wedge \ \top$ に SAT ソルバーを起動すると，$(x_1, x_2, x_3) = (1, 1, 0)$ がモデル M として求められる.

2. 1. で求めたモデル M から次の F1，F2 を作る

$$\begin{aligned} \mathrm{F1} &= \neg(x_1 \wedge x_2) \\ \mathrm{F2} &= \neg x_3 \end{aligned}$$

3. $P \coloneqq P \wedge \neg(x_1 \wedge x_2) \wedge \neg x_3$ として，SAT ソルバーを起動すると，UNSAT であるので $(x_1, x_2, x_3) = (1, 1, 0)$ は極小モデルである.

以上のようにして，極小モデルが求まる.
ただし，上の例の 1. で求まるモデル M はいくつかあり，あくまで一例である. 上のような操作を繰り返し行うことで，全ての極小モデルを列挙することができる.
また，先ほどの手順 2. における F1，F2 を以下のように変更すれば，極大モデルを求めることができる.

$$\text{F1} = \neg(\neg y_1 \wedge ... \wedge \neg y_n)$$
$$\text{F2} = x_1 \wedge ... \wedge x_m$$

極小モデルでは，いま見つけたモデル M より真となるリテラルが少ないモデルが存在するかを SAT ソルバーで調べ、存在しなければモデル M を極小モデルとして返すという計算方法であったので，極大モデルでは，いま見つけたモデル M より偽となるリテラルが少ないモデルが存在するかを SAT ソルバーで調べ、存在しなければモデル M を極大モデルとして返すという計算方法にしている.

第4章　SAT符号化を用いたSATソルバー1回起動による極小・極大モデル計算方法

3章では，SATソルバーを複数回起動することによって極小・極大モデルを計算する方法を紹介した. ただ，SATソルバーを複数回起動するのは手間がかかってしまう. 論文[10]では，SAT問題に符号化を施したCNFに変換を施して，SATソルバーを1回起動することで，極小モデルを計算する方法が提案されたが，計算方法の実装や評価は示されていなかった. そこで，本論文では，SATソルバー1回の起動で極大モデルを計算できるようなCNFへの変換方法を提案し，極小モデルの計算方法とともに実装，評価する. また，本章ではSAT問題に符号化を施したCNFに対して変換を施し，SATソルバーを1回起動することで，極小・極大モデルを計算できる変換方法について説明する.

4.1　SATソルバー1回起動による極小モデル計算の先行研究

極小モデル計算のためのCNF変換方法

本節では，論文[10]で提案された，SAT問題に符号化を施したCNFの正リテラルに注目し，できる限り真となる変数を減らしたいという動機のもとCNFに新たな制約を追加し，SATソルバー1回起動によって極小モデルを計算する手法について説明する. この計算手法では，ある節中の他の変数で節が満たされている時，その変数を偽にするという制約を追加し，極小モデルを求める.

CNFの変換式

与えられたCNFをΦ，変換後のCNFをΩ_{min}とすると，提案変換は，Φに以下のような制約$M_{min}(\Phi)$を追加し，$\Omega_{min} = \Phi \wedge M_{min}(\Phi)$とするものである．

式 1

$$M_{min}(\Phi) \equiv \bigwedge_{x \in Var(\Phi)} Cl_{min}(\Phi, x) \rightarrow \neg x$$
$$Cl_{min}(\Phi, x) \equiv \bigwedge_{c \in \Phi, x \in c} c \backslash \{x\}$$

ここで，$Cl_{min}(\Phi, x)$は，変数xが正リテラルとして含まれる全ての節からxを除いたものの連言であり，$M_{min}(\Phi)$は全ての変数xにおいて，$Cl_{min}(\Phi, x)$が満たされるならば，xを偽とするという制約の連言である．

例 5 以下のようなCNF式Φが与えられているとする．

$$\Phi \ = \ (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_2 \vee \neg x_3)$$

このとき，$Cl_{min}(\Phi, x_1)$は，各節から正リテラルとして含まれる変数x_1を除いたものの連言であるので，以下のようになる．

$$Cl_{min}(\Phi, x_1) \ = \ \neg x_2 \wedge (x_2 \vee \neg x_3)$$

すると，$M_{min}(\Phi)$により，$Cl_{min}(\Phi, x_1)$が満たされるならば，x_1を偽とするという以下の制約が追加される．

$$(\neg x_2 \wedge (x_2 \vee \neg x_3)) \rightarrow \neg x_1$$

この制約は，$(x_2, x_3) \ = \ (0, 0)$の時，x_1は偽になるということを意味する．

4.2　SATソルバー1回起動による極大モデル計算方法

極大モデル計算のためのCNF変換方法

4.1節では，SAT問題に符号化を施したCNFの正リテラルに注目し，できる限り真となる変数を減らしたいという動機のもとCNFに新たな制約

を追加し，極小モデルを計算した. 本節では，SAT 問題に符号化を施した CNF の負リテラルに注目し，できる限り偽となる変数を減らしたいという動機のもと CNF に新たな制約を追加し，極大モデルを計算する方法を提案する. ある節中の他の変数で節が満たされている時，その変数を真にするというのが変換によって追加される制約の意図である.

CNF の変換式

与えられた CNF を Φ，変換後の CNF を Ω_{max} とすると，提案変換は，Φ に以下のような制約 $M_{max}(\Phi)$ を追加し，$\Omega_{max} = \Phi \wedge M_{max}(\Phi)$ とするものである.

式 2

$$M_{max}(\Phi) \equiv \bigwedge_{x \in Var(\Phi)} Cl_{max}(\Phi, \neg x) \rightarrow x$$
$$Cl_{max}(\Phi, \neg x) \equiv \bigwedge_{c \in \Phi, \neg x \in c} c \backslash \{\neg x\}$$

ここで，$Cl_{max}(\Phi, \neg x)$ は，変数 x が負リテラルとして含まれる全ての節から $\neg x$ を除いたものの連言であり，$M_{max}(\Phi)$ は全ての変数 x において，$Cl_{max}(\Phi, \neg x)$ が満たされるならば，x を真とするという制約の連言である.

例 6 以下のような CNF 式 Φ が与えられているとする.

$$\Phi \ = \ (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

このとき，$Cl_{max}(\Phi, \neg x_1)$ は，各節から負リテラルとして含まれる変数 x_1 を除いたものの連言であるので，以下のようになる.

$$Cl_{max}(\Phi, \neg x_1) \ = \ x_2 \wedge (\neg x_2 \vee x_3)$$

すると，$M_{max}(\Phi)$ により，$Cl_{max}(\Phi, \neg x_1)$ が満たされるならば,x_1 を真とするという以下の制約が追加される.

$$(x_2 \wedge (\neg x_2 \vee x_3)) \rightarrow x_1$$

この制約は，$(x_2, x_3) \ = \ (1, 1)$ の時，x_1 は真になるということを意味する.

極小モデル，極大モデル計算の例と計算過程

例 7 (極小モデル計算の例) 以下のようなCNF式Φが与えられているとする.

$$\Phi \ = \ (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3) \wedge (x_3 \vee \neg x_4)$$

このCNFΦのモデルは以下のようになる.

$$(x_1, x_2, x_3, x_4) = (1,0,1,0),\ (1,1,1,0),\ (0,0,1,0),\ (0,1,1,0),\\ (1,0,1,1),\ (0,0,1,1),\ (1,1,1,1),\ (0,1,1,1),\ (1,1,0,0)$$

このCNFΦに式1に沿って変換を施すと，表4.1のような制約が追加され，最終的に$(x_1, x_2, x_3, x_4) \ = \ (0,0,1,0), (1,1,0,0)$という極小モデルが計算でき，これは$\Phi$の極小モデルに一致する.

表 4.1: CNFΦの極小変換

注目変数	追加する節（制約）	制限されるモデル	極小モデル候補
x_1	$(\neg x_2 \vee x_3) \rightarrow \neg x_1$	$(1,0,1,0), (1,1,1,0),$ $(1,0,1,1), (1,1,1,1)$	$(0,0,1,0), (0,1,1,0),$ $(0,0,1,1), (0,1,1,1),$ $(1,1,0,0)$
x_2	$x_3 \rightarrow \neg x_2$	$(1,1,1,0), (0,1,1,0),$ $(1,1,1,1), (0,1,1,1)$	$(0,0,1,0), (0,0,1,1),$ $(1,1,0,0)$
x_3	$((x_1 \vee \neg x_2) \wedge x_2 \wedge \neg x_4)$ $\rightarrow \neg x_3$	$(1,1,1,0)$	$(0,0,1,0), (0,0,1,1),$ $(1,1,0,0)$
x_4	$\top \rightarrow \neg x_4 \ (\equiv \neg x_4)$	$(1,0,1,1), (0,0,1,1),$ $(1,1,1,1), (0,1,1,1)$	$(0,0,1,0), (1,1,0,0)$

表4.1で行われた極小モデル計算をさらに詳しく解説する. まず変数x_1に注目すると，1つ目の節で正リテラルとして出現しているため，変数x_1以外の変数で他の節が真である時，変数x_1が偽となる制約$(\neg x_2 \wedge x_3) \rightarrow \neg x_1$を追加する．次に変数$x_2$に注目すると，2つ目の節で正リテラルとして出現しているため，変数x_2以外の変数で他の節が真である時，変数x_2が偽となる制約$x_3 \ \rightarrow \ \neg x_2$を追加する．次に変数$x_3$に注目すると，すべての節で正リテラルとして出現している．このとき，全ての節において，変数x_3以外の変数で節が真である時，変数x_3が偽となる制約

$((x_1 \vee \neg x_2) \wedge x_2 \wedge \neg x_4) \rightarrow \neg x_3$ を追加する．次に変数 x_4 に注目すると，変数 x_4 を正リテラルとして含む節は存在しない．すなわち，x_4 が偽であっても全ての節が充足されないことはないため，単位節 $\neg x_4$ を追加する．このようにして，CNF に変換を加え，新たな制約を加えることで極小モデルを計算することができる．

例 8 (極大モデル計算の例) 以下のような CNF 式 Φ が与えられているとする．

$$\Phi \ = \ (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_4)$$

この CNFΦ のモデルは以下のようになる．

$$(x_1, x_2, x_3, x_4) = (0,0,0,0),\ (0,0,0,1),\ (0,0,1,1),\ (0,1,0,0),\\ (0,1,0,1),\ (1,0,0,0),\ (1,0,0,1),\ (1,1,0,0),\ (1,1,0,1)$$

この CNFΦ に式 2 に沿って変換を施すと，表 4.2 のような制約が追加され，最終的に $(x_1, x_2, x_3, x_4) \ = \ (0,0,1,1), (1,1,0,1)$ という極大モデルが計算でき，これらは Φ の極大モデルに一致する．

表 4.2: CNFΦ の極大変換

注目変数	追加する節（制約）	制限されるモデル	極大モデル候補
x_1	$(x_2 \vee \neg x_3) \rightarrow x_1$	$(0,0,0,0), (0,0,0,1),$ $(0,1,0,0), (0,1,0,1)$	$(0,0,1,1), (1,0,0,0),$ $(1,0,0,1), (1,1,0,0),$ $(1,1,0,1)$
x_2	$\neg x_3 \rightarrow x_2$	$(0,0,0,0), (0,0,0,1),$ $(1,0,0,0), (1,0,0,1)$	$(0,0,1,1), (1,1,0,0),$ $(1,1,0,1)$
x_3	$((\neg x_1 \vee x_2) \wedge \neg x_2 \wedge x_4)$ $\rightarrow x_3$	$(0,0,0,1)$	$(0,0,1,1), (1,1,0,0),$ $(1,1,0,1)$
x_4	$\top \rightarrow x_4 \ (\equiv x_4)$	$(0,0,0,0), (0,1,0,0),$ $(1,0,0,0), (1,1,0,0)$	$(0,0,1,1), (1,1,0,1)$

表 4.2 で行われた極大モデル計算をさらに詳しく解説する．まず変数 x_1 に注目すると，1 つ目の節で負リテラルとして出現しているため，変数 x_1 以外の変数で他の節が真である時，変数 x_1 が真となる制約 $(x_2 \wedge \neg x_3) \rightarrow x_1$ を追加する．次に変数 x_2 に注目すると，2 つ目の節で負リテラルとして出現し

ているため，変数x_2以外の変数で他の節が真である時，変数x_2が真となる制約$\neg x_3 \to x_2$を追加する．次に変数x_3に注目すると，すべての節で負リテラルとして出現している．このとき，全ての節において，変数x_3以外の変数で節が真である時，変数x_3が真となる制約$((\neg x_1 \vee x_2) \wedge \neg x_2 \wedge x_4) \to x_3$を追加する．次に変数$x_4$に注目すると，変数$x_4$を負リテラルとして含む節は存在しない．すなわち，$x_4$が真であっても全ての節が充足されないことはないため，単位節x_4を追加する．
このようにして，CNF に変換を加え，新たな制約を加えることで極大モデルを計算することができる．

第5章　提案手法の実装

5.1　変換アルゴリズム

極小変換，極大変換の一般的なアルゴリズムを Algorithm2 に記述する．アルゴリズムでは，入力を CNFΦ と min,max とし，入力によって極小変換と極大変換に分岐できる．また出力は変換後の CNFΩ である．

CNF Φ から変換後の CNF Ω を構成する手順を，行番号に沿って説明する．Algorithm 2 は入力として CNF Φ と mode $\in \{\min, \max\}$ を受け取り，補助制約 M を生成したうえで $\Omega \leftarrow \Phi \wedge M$ を返す（行 1–2, 28–29）．

準備（行 1–2）

まず Φ に出現する命題変数全体を $P \leftarrow \mathrm{Var}(\Phi)$ として取り出す（行 1）．続いて，追加制約を保管するための CNF を $M \leftarrow \top$ と初期化する（行 2）．

mode による分岐（行 3–11）

各変数 $p \in P$ について，その変数に関する「極小／極大らしさ」を強制する制約を追加する（行 3）．このとき，まず $c_1 \leftarrow \bot$ を用意し（行 4），mode に応じて 2 つのリテラル t と h を次のように選ぶ（行 5–11）．

- mode = min のとき： $t \leftarrow p,\ \ h \leftarrow \neg p$（行 6–7）．
- mode = max のとき： $t \leftarrow \neg p,\ \ h \leftarrow p$（行 9–10）．

ここで t は「節の中で注目するリテラル」，h は最終的に追加する含意 $(h \vee c_1)$ の左側に出るリテラル（行 26）である．この入れ替えにより，極小変換と極大変換を同一の枠組みで処理できる．

各節に対する補助変数の導入（行 12–25）

次に Φ の各節 $clause \in \Phi$ を走査し（行 12），その節が t を含む場合のみ処理を行う（行 13）.
$t \in clause$ であるとき，新しい補助変数 x を 1 つ導入し（行 14），さらに $c_2 \leftarrow \bot$ を初期化する（行 15）.
その後，節 $clause$ 内の各リテラル $l \in clause$ を走査し（行 16），$l \neq t$ のものだけを用いて以下を行う（行 17–20）.

1. 制約 $(\neg x \vee \neg l)$ を M に追加する（行 18）.
2. $c_2 \leftarrow c_2 \vee l$ として，t 以外のリテラルの選言 c_2 を作る（行 19）.

このとき，$(\neg x \vee \neg l)$ は $x \rightarrow \neg l$ を表すため，x が真なら「t 以外のリテラルはすべて偽である」ことを要求する.

ループ後（行 21），さらに $(c_2 \vee x)$ を M に追加する（行 22）．これは $(\neg c_2) \rightarrow x$ を表し，節中の他リテラル（t 以外）がすべて偽（すなわち c_2 が偽）なら x を真にせよ，という意味になる．結果として，x は

$$x \;\leftrightarrow \bigwedge_{l \in clause,\ l \neq t} \neg l$$

（「その節で t 以外がすべて偽である」こと）を CNF で表す指示変数として機能する．最後に $c_1 \leftarrow c_1 \vee x$ として（行 23），t を含む各節について生成した x の選言を c_1 に保管する（行 23）.

変数ごとの主要制約の追加（行 26）

全節の走査が終わると，$M \leftarrow M \wedge (h \vee c_1)$ を追加する（行 26）.
c_1 は「t を含むどれかの節で，t 以外がすべて偽になる（＝その節が t に依存する）」ことを表す．したがって $(h \vee c_1)$ は

$$\neg h \rightarrow c_1$$

を意味し，**h を偽にする（min なら p を真にする，max なら p を偽にする）場合には，その選択が必須である状況（c_1 が真）を要求する**という形で，極小性，極大性を CNF 制約として埋め込む.

出力（行 28–29）

以上で得た M を元の CNF に連言し，$\Omega \leftarrow \Phi \wedge M$ を返す（行 28–29）．この Ω を SAT ソルバーで 1 回解くことで，mode に応じた極小モデル，極大モデルに対応する解を得る（Algorithm 2 の目的）．

5.2 変換アルゴリズムの動作例

本節では，Algorithm 2 が具体的にどのように新規変数を追加し，CNF を変換するかを，小さなサイズの CNF を入力として示す．

入力 CNF

次の CNF を入力 Φ とする：

$$\Phi \ = (p \vee q) \ \wedge \ (p \vee \neg q) \ \wedge \ (\neg p \vee r). \tag{5.1}$$

出現変数は $P = \mathrm{Var}(\Phi) = \{p, q, r\}$ である．

極小変換（mode=min)

Algorithm 2 は各 $p \in P$ について，t を含む節ごとに新規変数を導入し，最後に $(h \vee c_1)$ を追加する．以下では，変数ごとに追加される新規変数と制約を明示する．
また，行 26 で各変数 p に対して追加する節 $(h \vee c_1)$ を以後，極小（極大）性制約と呼ぶ．

(1) 変数 p に対する処理（min：$t = p,\ h = \neg p$）

Φ のうち $t = p$ を含む節は

$$C_1 = (p \vee q)$$
$$C_2 = (p \vee \neg q)$$

の 2 つである．それぞれに対して新規変数を導入する

- 節 $C_1 = (p \vee q)$ に対して新規変数 $x_{p,1}$ を導入する．この節で $t = p$ 以外のリテラルは q だけなので，$x_{p,1}$ は q が偽であることを表す指示変数（$x_{p,1} \leftrightarrow \neg q$）になる．Algorithm 2（行 18,22）に従い，次を M に追加する

$$(\neg x_{p,1} \vee \neg q) \ \wedge \ (q \vee x_{p,1}). \tag{5.2}$$

- 節 $C_2 = (p \vee \neg q)$ に対して新規変数 $x_{p,2}$ を導入する．$t = p$ 以外のリテラルは $\neg q$ だけなので，$x_{p,2} \leftrightarrow \neg(\neg q) \equiv q$ を表す（$x_{p,2} \leftrightarrow q$）．よって追加制約は

$$(\neg x_{p,2} \vee \neg(\neg q)) \ \wedge \ (\neg q \vee x_{p,2}) \ \equiv \ (\neg x_{p,2} \vee q) \ \wedge \ (\neg q \vee x_{p,2}). \tag{5.3}$$

このとき，c_1 は，p を含むどれかの節で，p 以外がすべて偽を表す選言なので

$$c_1 = x_{p,1} \ \vee \ x_{p,2}.$$

最後に Algorithm 2（行 26）により極小性制約 $(h \vee c_1)$ を追加する

$$(\neg p \vee x_{p,1} \vee x_{p,2}). \tag{5.4}$$

(2) 変数 q に対する処理（min：$t = q,\ h = \neg q$）

今度は $t = q$ を含む節だけを見る．Φ で q を含むのは

$$C_1 = (p \vee q)$$

のみである．したがって新規変数は 1 つ

- 節 $C_1 = (p \vee q)$ に対して新規変数 $x_{q,1}$ を導入する．$t = q$ 以外のリテラルは p だけなので，$x_{q,1} \leftrightarrow \neg p$ を表す．追加制約は

$$(\neg x_{q,1} \vee \neg p) \ \wedge \ (p \vee x_{q,1}). \tag{5.5}$$

このとき $c_1 = x_{q,1}$ であり，極小性制約は

$$(\neg q \vee x_{q,1}). \tag{5.6}$$

(3) 変数 r に対する処理（min：$t = r,\ h = \neg r$）

Φ で r を含むのは

$$C_3 = (\neg p \vee r)$$

のみである．新規変数を 1 つ導入する：

- 節 $C_3 = (\neg p \vee r)$ に対して新規変数 $x_{r,1}$ を導入する．$t = r$ 以外のリテラルは $\neg p$ なので，$x_{r,1} \leftrightarrow \neg(\neg p) \equiv p$ を表す．追加制約は：

$$(\neg x_{r,1} \vee \neg(\neg p)) \ \wedge \ (\neg p \vee x_{r,1}) \ \equiv \ (\neg x_{r,1} \vee p) \ \wedge \ (\neg p \vee x_{r,1}). \tag{5.7}$$

このとき $c_1 = x_{r,1}$ であり，極小性制約は：

$$(\neg r \vee x_{r,1}). \tag{5.8}$$

得られる追加制約 M と出力 Ω

以上で生成された追加制約 M は

$$\begin{aligned} M \ = & (\neg x_{p,1} \vee \neg q) \wedge (q \vee x_{p,1}) \wedge (\neg x_{p,2} \vee q) \wedge (\neg q \vee x_{p,2}) \wedge (\neg p \vee x_{p,1} \vee x_{p,2}) \\ & \wedge (\neg x_{q,1} \vee \neg p) \wedge (p \vee x_{q,1}) \wedge (\neg q \vee x_{q,1}) \\ & \wedge (\neg x_{r,1} \vee p) \wedge (\neg p \vee x_{r,1}) \wedge (\neg r \vee x_{r,1}). \end{aligned} \tag{5.9}$$

したがって出力は $\Omega = \Phi \wedge M$ である：

$$\Omega \ = (p \vee q) \wedge (p \vee \neg q) \wedge (\neg p \vee r) \ \wedge \ M. \tag{5.10}$$

導入された各新規変数は，対応する節において，注目リテラル t 以外がすべて偽を表す指示変数である（Algorithm 2 行 18–22）．例えば

$$x_{p,1} \leftrightarrow \neg q, \qquad x_{p,2} \leftrightarrow q, \qquad x_{q,1} \leftrightarrow \neg p, \qquad x_{r,1} \leftrightarrow p$$

となっており，極小性制約

$$(\neg p \vee x_{p,1} \vee x_{p,2}), \quad (\neg q \vee x_{q,1}), \quad (\neg r \vee x_{r,1})$$

は p, q, r を真にする場合は，それが必須であることを x によって証明しなければならないという意味を持ち，CNF に埋め込むことで極小性を与える．

極大変換（mode=max ）

極大変換では，各変数ごとに

$$t \leftarrow \neg p, \quad h \leftarrow p$$

のように t と h が入れ替わるだけで，節の走査・新規変数の導入・制約生成は同一である（Algorithm 2 行 5–11）．したがって，上の例をそのまま用い，$\neg p$ を含む節だけが p の処理対象になる点に注意して同様に展開できる．

Algorithm 2 CNF Φ から CNF への変換方法 Ω(min/max)

Input: CNF Φ, mode $\in \{\min, \max\}$
Output: CNF Ω

$P \leftarrow \mathrm{Var}(\Phi)$
$M \leftarrow \top$
for all $p \in P$ **do**
    $c_1 \leftarrow \bot$
    if $mode = \min$ **then**
        $t \leftarrow p$
        $h \leftarrow \neg p$
    else
        $t \leftarrow \neg p$
        $h \leftarrow p$
    end if
    for all $clause \in \Phi$ **do**
        if $t \in clause$ **then**
            $x \leftarrow$ new variable
            $c_2 \leftarrow \bot$
            for all $l \in clause$ **do**
                if $l \neq t$ **then**
                    $M \leftarrow M \wedge (\neg x \vee \neg l)$
                    $c_2 \leftarrow c_2 \vee l$
                end if
            end for
            $M \leftarrow M \wedge (c_2 \vee x)$
            $c_1 \leftarrow c_1 \vee x$
        end if
    end for
    $M \leftarrow M \wedge (h \vee c_1)$
end for
$\Omega \leftarrow \Phi \wedge M$
return Ω

第6章　実行実験

6.1　実験概要

論文 [10] では，SAT ソルバーを 1 回起動するだけで極小モデルを計算する手法が提案されたが，当該手法の実行性能に関する評価は十分に行われていなかった. 同様に，本論文で提案する，SAT ソルバー 1 回起動による極大モデル計算方法についても，その性能は未評価である. そこで本節では，極小モデル計算の評価対象としてグラフ問題である極小支配集合問題（Minimal Dominating Set: MDS）を，極大モデル計算の評価対象としてグラフ問題である極大独立集合問題（Maximal Independent Set: MIS）を用いる. 具体的には，論文 [4] で提案された極大モデル計算手法（MIS-basic）および極小モデル計算手法（MDS-basic），論文 [10] で提案された極小モデル計算手法（MDS-trans），ならびに本論文で提案する極大モデル計算手法（MIS-trans）を比較対象とし，これらの手法に対して実行実験を行うことで性能評価を行う.

6.2　実験環境

本実験における実行環境は以下の通りである.

- 実行環境：Mac mini，Apple M4，32GB
- SAT ソルバー：*CaDiCaL* 3.0.0
- 使用言語：*Rust*
- 制限時間：1 問当たり 30 分

6.3　ベンチマーク

今回の実行実験で使用するベンチマークは 3 行 n 列（$n = 1, 2, ..., 19$）の Grid graph の全 19 問である．ここで，Grid graph とは，整数格子上の点を頂点とし，上下左右に隣接する点同士を辺で結んだグラフである．
支配集合とは，グラフ $G = (V, E)$ において，頂点集合 $D \subseteq V$ が，全ての頂点を自分自身または隣接頂点で覆うとき，D を支配集合と呼ぶ．すなわち，任意の頂点 $v \in V$ について

$$v \in D \quad \text{または} \quad \exists u \in D \text{ s.t. } \{u, v\} \in E$$

が成り立つ．
独立集合とは，グラフ $G = (V, E)$ において，頂点集合 $I \subseteq V$ が集合内の頂点同士が隣り合わない性質を満たすとき，I を独立集合と呼ぶ．これは，

$$\forall u, v \in I \ (u \neq v) \Rightarrow \{u, v\} \notin E$$

で表される．

例 9 図 6.2 に 3×4 の Grid graph における極小支配集合（MDS）の例を，図 6.1 に 3×4 の Grid graph における極大独立集合（MIS) の例を示す．

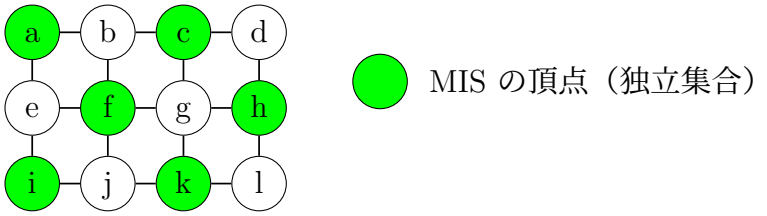

図 6.1: 3×4Grid graph の MIS

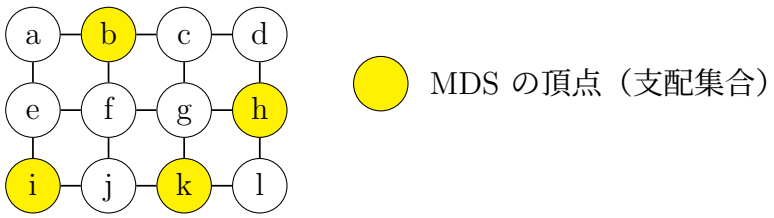

図 6.2: 3×4Grid graph の MDS

6.4 CPU 時間の比較

表 6.1 に MDS の既存手法と提案手法，表 6.2 に MIS の既存手法と提案手法の実行結果（各ベンチマークに対する CPU 時間と得られた結果数）を示す．左に既存手法の CPU 時間（秒）と得られた結果数，右に提案手法の CPU 時間（秒）と得られた結果数を示す．各行において同一ベンチマークに対する両手法の計算時間結果を比較できる．また，赤文字で示されている結果は既存手法より短い時間で解けた問題である.MDS および MIS について，それぞれのカクタスプロットを図 6.3，図 6.4 に示す．横軸は Grid graph の列数 n，縦軸は各問題を解くのに要した CPU 時間である．同一の縦軸値に対して右側に位置するほど多くの問題を解けたことを意味し，下側に位置するほど短時間で解けたことを意味する．

図 6.5 に SAT ソルバー起動に要した CPU 時間とモデル計算に要した CPU 時間の比較を示す．縦軸は CPU 時間（秒），横軸は計算したモデルの数を示している. また，緑のプロットは既存手法でモデル計算に要した合計時間，赤のプロットは提案手法でモデル計算に要した合計時間，水色の棒グラフは既存手法において SAT ソルバー起動でモデルを発見するために要した時間，青の棒グラフは既存手法において SAT ソルバー起動で UNSAT を出力するために要した時間である.

図 6.6 に既存手法と提案手法のモデル計算に要した CPU 時間推移の比較を示す．青のグラフが既存手法において各モデル計算に要した時間を，赤のグラフが提案手法において各モデル計算に要した時間を示している．このグラフでは上に行くほどモデル計算に時間を要したと言える.

まず表 6.1 より，極小支配集合問題の全解列挙において既存手法は 3×10 まで解けたのに対し，提案手法は 3×12 まで解けた．さらに同一サイズの問題で比較すると，3×10 において既存手法が 1361.24 秒を要したのに対し，提案手法は 3.620 秒で解けており，20 分以上の大幅な短縮が確認できる．

次に表 6.2 より，極大独立集合問題の全解列挙において既存手法は 3×13 まで解けたのに対し，提案手法は 3×18 まで解けた．また 3×13 における CPU 時間は，既存手法が 584.70 秒であるのに対し，提案手法は 0.36 秒であり，約 10 分の短縮が確認できる．

図 6.3 および図 6.4 においても，いずれの問題設定でも提案手法が既存手法よりも下側かつ右側に位置している．したがって，提案手法は既存手法に比べて，同一問題をより短時間で解けるだけでなく，制限時間内

に解ける問題サイズの上限も拡大しており，総合的に高い性能を示すことが分かる．

また，図 6.5 より，MDS および MIS のいずれにおいても，既存手法では頻繁な SAT ソルバーの起動がオーバーヘッドとなり，CPU 時間全体の増加を招いていることが分かる．一方，提案手法ではソルバーの起動回数が大幅に抑制されているため，起動に伴うコストが低減され，効率的な計算が実現できている．

さらに，図 6.6 より，提案手法は既存手法と比較して，各モデルを計算するのに要する時間がモデル数の増加に伴い短くなり，かつ安定して推移していることが読み取れる．これは，提案手法が大規模な解空間を持つ問題に対しても，高速に解を列挙し続けられることを示しており，CPU 時間の短縮を裏付けるものである．

表 6.1: MDS の CPU 時間と結果数

Benchmark	MDS（既存）		MDS（提案）	
	time	results	time	results
grid3×1	< 0.0	2	< 0.0	2
grid3×2	< 0.0	7	< 0.0	7
grid3×3	< 0.0	16	< 0.0	16
grid3×4	< 0.0	53	< 0.0	53
grid3×5	0.01	154	< 0.0	154
grid3×6	0.15	436	0.01	436
grid3×7	1.86	1268	0.03	1268
grid3×8	26.32	3660	0.13	3660
grid3×9	182.35	10610	0.58	10610
grid3×10	1361.24	30744	3.620	30744
grid3×11	T.O.	-	24.84	89079
grid3×12	T.O.	-	207.90	258251
grid3×13	T.O.	-	T.O.	-

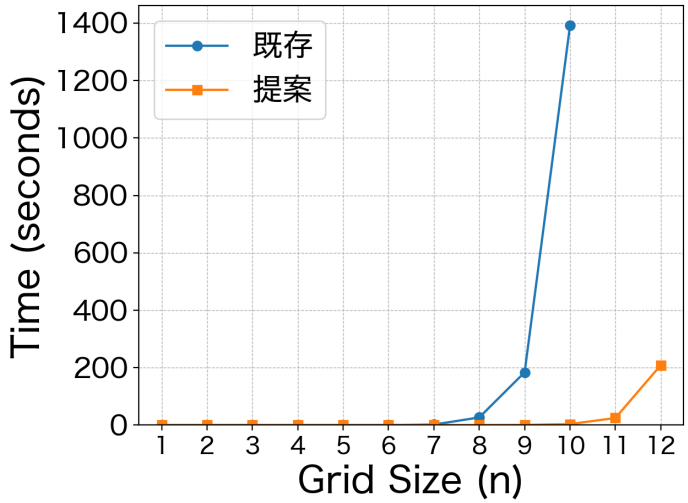

図 6.3: MDS の CPU 時間と結果数のプロット

表 6.2: MIS の CPU 時間と結果数

Benchmark	MIS（既存）		MIS（提案）	
	time	results	time	results
grid3×1	< 0.0	2	< 0.0	2
grid3×2	< 0.0	4	< 0.0	4
grid3×3	< 0.0	10	< 0.0	10
grid3×4	< 0.0	18	< 0.0	18
grid3×5	< 0.0	38	< 0.0	38
grid3×6	< 0.0	74	< 0.0	74
grid3×7	0.01	148	< 0.0	148
grid3×8	0.01	290	< 0.0	290
grid3×9	0.04	578	0.01	578
grid3×10	0.14	1156	0.01	1156
grid3×11	0.36	2706	0.05	2706
grid3×12	114.72	5518	0.14	5518
grid3×13	584.70	11228	0.36	11228
grid3×14	T.O.	-	1.52	22884
grid3×15	T.O.	-	5.26	46634
grid3×16	T.O.	-	23.53	94978
grid3×17	T.O.	-	110.72	193518
grid3×18	T.O.	-	756.59	394286
grid3×19	T.O.	-	T.O.	-

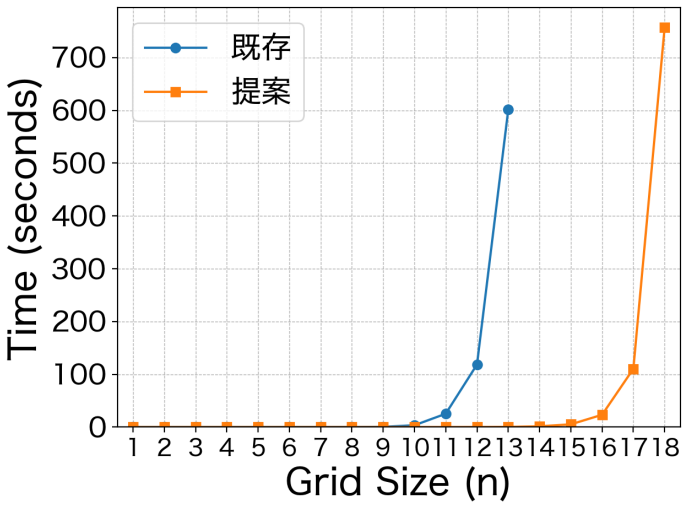

図 6.4: MIS の CPU 時間と結果数のプロット

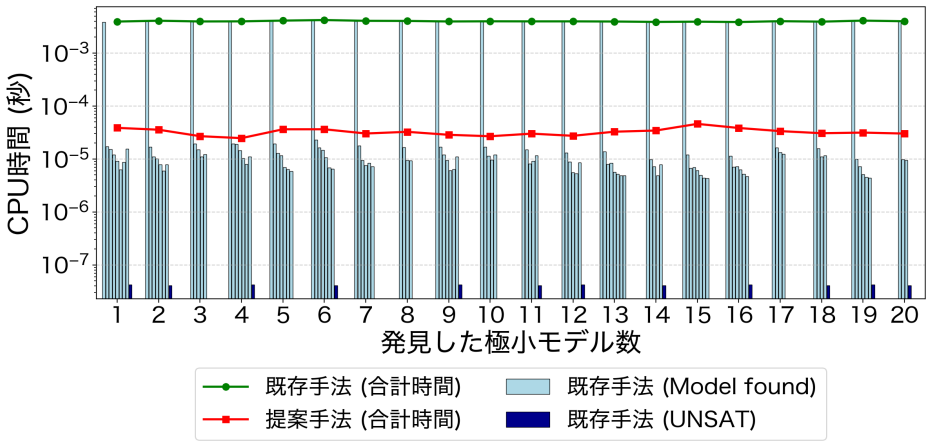

図 6.5: SAT ソルバー起動により要した CPU 時間の比較

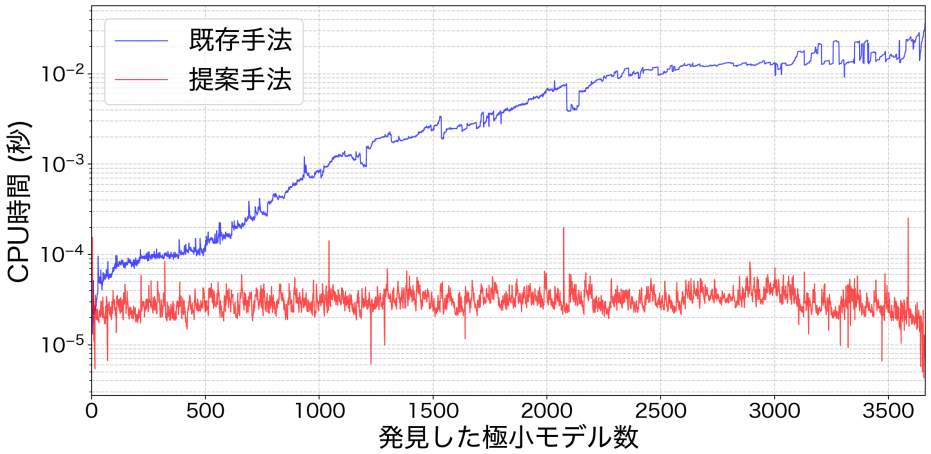

図 6.6: 既存手法と提案手法の計算したモデル数と CPU 時間推移

6.5　CNF の変数，節の数と SAT ソルバー起動回数の比較

表 6.3 に，Grid graph を CNF として定式化した際の変数数と節数を示す．vars が変数の数，clauses が節の数を示している. また，提案手法では CNF を変換した後の変数，節の数が示されている. 各行において同一ベンチマークに対する両手法の変数，節の数を比較できる．

表 6.4 に，各計算手法における SAT ソルバー（CaDiCaL）の起動回数と，既存手法に対する提案手法の割合（%) を示す．各行において同一ベンチマークに対する両手法の SAT ソルバー起動回数を比較できる．

表 6.3 より，MDS での提案手法では，CNF の変換過程で Tseitin 変換を用いているため，補助変数が導入され，既存手法に比べて変数数が増加している．MIS の方が既存手法と提案手法の性能差が大きく現れている要因として，MDS での提案手法では Tseitin 変換により CNF の構造が複雑化することで CPU 時間が増加しやすい点が挙げられる．その結果として，MDS では変換による CPU 時間の高速化が MIS ほど顕著に表れにくいと考えられる．

表 6.4 より，提案手法では，SAT ソルバーの起動回数が期待通り，得られた解の数と一致しており，解の列挙に対して無駄な起動が発生していないことが確認できる．具体例として 3×8 の Grid graph に着目すると，MDS の既存手法は計算過程で CaDiCaL を 17406 回起動しているのに対し，提案手法は 3660 回であり，提案手法は既存手法に比べて約 21% の起動回数で済んでいる．このように SAT ソルバー起動回数が減少していることは，ソルバー起動に伴うオーバーヘッドの削減につながるため，CPU 時間の短縮に寄与していると考えられる．

表 6.3: MDS と MIS の CNF の変数数と節数の比較

	MDS（極小モデル）				MIS（極大モデル）			
	既存		提案		既存		提案	
Benchmark	vars	clauses	vars	clauses	vars	clauses	vars	clauses
grid3×1	3	3	10	16	3	2	3	5
grid3×2	6	6	26	60	6	7	6	13
grid3×3	9	9	42	110	9	12	9	21
grid3×4	12	12	58	160	12	17	12	29
grid3×5	15	15	74	210	15	22	15	37
grid3×6	18	18	90	260	18	27	18	45
grid3×7	21	21	106	310	21	32	21	53
grid3×8	24	24	122	360	24	37	24	61
grid3×9	27	27	138	410	27	42	27	69
grid3×10	30	30	154	460	30	47	30	77
grid3×11	-	-	170	510	33	52	33	85
grid3×12	-	-	186	560	36	57	36	93
grid3×13	-	-	-	-	39	62	39	101
grid3×14	-	-	-	-	-	-	42	109
grid3×15	-	-	-	-	-	-	45	117
grid3×16	-	-	-	-	-	-	48	125
grid3×17	-	-	-	-	-	-	51	133
grid3×18	-	-	-	-	-	-	54	141
grid3×19	-	-	-	-	-	-	-	-

表 6.4: MDS と MIS の SAT ソルバー起動回数の比較

	MDS（極小モデル）			MIS（極大モデル）		
Benchmark	既存	提案	割合 [%]	既存	提案	割合 [%]
grid3×1	7	2	28.6	6	2	33.3
grid3×2	23	7	30.4	11	4	36.4
grid3×3	62	16	25.8	23	10	43.5
grid3×4	207	53	25.6	48	18	37.5
grid3×5	594	154	25.9	101	38	37.6
grid3×6	1743	436	25.0	203	78	38.5
grid3×7	5502	1268	23.0	440	156	35.5
grid3×8	17406	3660	21.0	907	321	35.3
grid3×9	49864	10610	21.3	1848	651	35.2
grid3×10	138754	30744	22.2	3808	1335	35.1
grid3×11	-	89079	-	7896	2706	34.2
grid3×12	-	258251	-	16127	5518	34.2
grid3×13	-	-	-	33281	11228	33.8
grid3×14	-	-	-	-	22884	-
grid3×15	-	-	-	-	46634	-
grid3×16	-	-	-	-	94978	-
grid3×17	-	-	-	-	193518	-
grid3×18	-	-	-	-	394286	-
grid3×19	-	-	-	-	-	-

第7章　おわりに

本論文では，SAT 符号化に基づく CNF 変換を用い，SAT ソルバーを 1 回起動するだけで命題論理式の極小モデル・極大モデルを計算する枠組みを整理した．極小モデルについては既存の変換手法を実装し，極大モデルについては負リテラルに着目した同様の発想による CNF 変換方法を新たに提案・実装した．

3 行 n 列の Grid graph に基づくベンチマークを用いて，提案手法（trans）と従来の反復起動法（basic）の性能を比較した．極小支配集合問題（MDS）では，basic が 3×10 までしか解けなかったのに対し，trans は 3×12 まで解くことができた．極大独立集合問題（MIS）では，basic が 3×13 までであったのに対し，trans は 3×18 まで解くことができた．いずれの問題設定においても，SAT ソルバーの起動回数が大幅に削減され，計算時間の短縮と解ける問題サイズの拡大が確認できた．以上の結果から，SAT 符号化による CNF 変換が極小・極大モデル計算の実用性を高める有効なアプローチであることを示した．

今後の課題として，まず，グラフ問題以外の制約充足問題や計画問題など，異なる問題領域に対しても同様の傾向が成り立つかを検証する必要がある．また，変換に伴う補助変数・節の増加による CNF サイズの増大が性能に与える影響を分析し，冗長制約の削減や Tseitin 変換の設計最適化などにより変換そのものを効率化する余地がある．さらに，CaDiCaL 以外のソルバーでの再現性確認や，特定の変数集合のみを極小化・極大化する部分集合最適化への拡張など，極小・極大モデル計算をより実用的なものに発展させていくことが期待される．

謝辞

本研究の機会を賜り，丁寧かつ熱心にご指導頂きました名古屋大学大学院情報学研究科 番原 睦則教授，宋剛秀准教授に深く感謝申し上げます．また，日頃より様々なご助言を下さいました番原研究室の皆様に感謝申し上げます．

参考文献

[1] Armin Biere. Bounded model checking. In *Handbook of Satisfiability*, pp. 457–481. IOS Press, 2009.

[2] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. The MIT Press, 2006.

[3] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI 1992)*, pp. 359–363, 1992.

[4] Miyuki Koshimura, Hidetomo Nabeshima, Hiroshi Fujita, and Ryuzo Hasegawa. Minimal model generation with respect to an atom set. *FTP*, Vol. 9, pp. 49–59, 2009.

[5] Ilkka Niemelä. A tableau calculus for minimal model reasoning. In *Theorem Proving with Analytic Tableaux and Related Methods*, Vol. 1071 of *Lecture Notes in Computer Science*, pp. 278–294. Springer, 1996.

[6] Takehide Soh, Mutsunori Banbara, Naoyuki Tamura, and Daniel Le Berre. Solving multiobjective discrete optimization problems with propositional minimal model generation. In J. Christopher Beck, editor, *Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings*, Vol. 10416 of *Lecture Notes in Computer Science*.

[7] Naoyuki Tamura and Mutsunori Banbara. Sugar: a CSP to SAT translator based on order encoding. In *Proceedings of the 2nd International CSP Solver Competition*, pp. 65–69, 2008.

[8] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear CSP into SAT. *Constraints*, Vol. 14, No. 2, pp. 254–272, 2009.

[9] 井上克巳, 田村直之. SAT ソルバーの基礎 (<特集>最近の SAT 技術の発展). 人工知能, Vol. 25, No. 1, pp. 57–67, 2010.

[10] 足立啓一. リテラルの充足に対する制約に基づく命題論理式の充足可能性を保つ変換に関する研究. 修士論文, 神戸大学大学院 システム情報学研究科, 2023.

[11] 長谷川隆三, 藤田博, 越村三幸. モデル列挙とモデル計数 (<特集>最近の SAT 技術の発展). 人工知能, Vol. 25, No. 1, pp. 96–104, 2010.