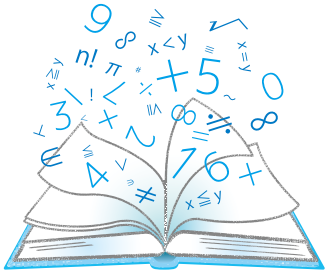


② SATとパズル

—問題をいかにSATソルバーで解くか—

田村直之（神戸大学） 宋 剛秀（神戸大学） 番原睦則（神戸大学）

SATソルバーはパズルを解く「銀の弾丸」？

本誌の読者には，数独などのパズルを解くソルバープログラムを書いた経験のある人も多いだろう．プログラミング技量を高める腕試しにもなるし，何より書いていて楽しい．

数独程度なら単純なバックトラック法で十分だが，後述のナンバーリンクなどは，効率の良い解法アルゴリズムを考えるのが難しい．また，小さいサイズの問題なら簡単に解けても，サイズが大きくなるにつれ急激に遅くなることも多い．これらのパズルの多くは，効率の良い解法アルゴリズムが知られていない種類の問題☆1なのだ．

本稿では，まったく違ったアプローチを紹介する．**SATソルバー**というプログラムを活用する方法だ[1),2)]．SATソルバーとは，与えられた命題論理式を満たす解を探すプログラムである．命題論理だから，式中に現れる変数（命題変数と呼ばれる）は真か偽（あるいは1か0）の2通りの値しかとらない．そんな単純なものを使って，難しい問題を解けるという．

このSATソルバーを使う手法は2003年頃にI教授から教えてもらった．初めて聞いたときの率直な印象は「コンピュータは0と1だけで動いているから，原理的にはSATソルバーで問題を解くことは可能だろう．しかし，実際に使えるのか？」だった．ところが，実際に使ってみて驚いた．100万個の命題変数を含む論理式でも苦もなく解いてしまう☆2．命題変数の個数をnとすると，調べるべき探索空間の大きさは2^nだから，とんでもない膨大な数の組合せを処理している．

図-1 Knuthの SAT 2012招待講演のスライド

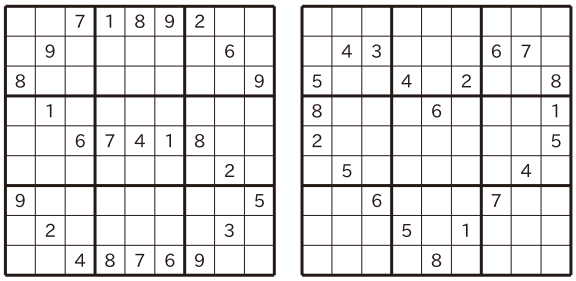
チューリング賞受賞者のKnuthによる有名な教科書“The Art of Computer Programming”の最新分冊では，300ページ以上もの分量がSATソルバーとその応用の説明に割かれている[3)]（500問以上の練習問題とその解答を含む）．Knuthは，その序文で「SAT問題は，非常に多くの問題を解くための鍵となることから，明らかにkiller appだ」と述べている．なお，KnuthのWebページからはSATソルバーを含むプログラムや**図-1**のスライドなどをダウンロードできる．

SATソルバーは「銀の弾丸」☆3なのだろうか．もちろん，解けない問題も数多く存在するし，専用プログラムのほうが速い場合もある．だが，多くの問題に対する有効な解決手段であることは間違いない．本稿では簡単なパズルを対象とし，SATソルバーを用いた問題解決方法について紹介する．

数独を解いてみよう

最初の題材として，海外でも有名な数独を取り上げよう．9つある各行，9つある各列，9つある

☆1 NP完全問題と呼ばれる．
☆2 もちろん，必ず解けるわけではない．
☆3 狼男や悪魔などを一発で倒すといわれる弾丸．困難な問題を直ちに解決する手段の比喩として用いられる．



(a) は http://puzzle.gr.jp で作成．(b) は藤原博文氏作

図 -2　数独の例

3 × 3 のブロックに 1 から 9 の数が 1 回ずつ出るように数を埋めるパズルだ．**図 -2** の (a) はヒントの数が 25 個の比較的簡単な問題，(b) はヒント数が 20 個で少し難しい問題である．

SAT ソルバーを利用する前段階として，まず数独の条件を数式で表現したもの（**制約モデル**と呼ばれる）を考えよう．

i 行目 j 列目のマスの値を x_{ij} で表すことにする．各 x_{ij} は 1 から 9 の値をとるから

$$x_{ij} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\} \tag{1}$$

と書ける．この条件は全体で $9 \times 9 = 81$ 個ある．

x_{ij} と x_{kl} が同一行（あるいは同一列，同一ブロック）にある場合，異なる値をとらなければならない．つまり，

$$x_{ij} \neq x_{kl} \tag{2}$$

と書ける．x_{ij} と同一行の x_{kl} は 8 個あり，同一列および同一ブロックもそれぞれ 8 個ある．重複を除けば合計で 20 個である．また $x_{ij} \neq x_{kl}$ か $x_{kl} \neq x_{ij}$ の一方で良いから，この条件は全体で 810 個の式になる．

最後に x_{ij} のマスにヒントとして数字 a が書かれている場合を考える．そのときは，

$$x_{ij} = a \tag{3}$$

を条件として加えれば良い．図 -2 (a) の場合，ヒント数と同じ 25 個の条件になる．

以上が数独の制約モデルであり，これを満たす解が数独の答えとなる．

★数独の SAT 符号化

SAT ソルバーを利用するには，制約モデルの各条件を命題論理式に変換しなければならない．このステップは **SAT 符号化**と呼ばれ，色々な方法が知られている．ここでは直接符号化と呼ばれる方法を使うことにしよう．

まず，命題変数 p_{ija} で x_{ij} の値が a に等しいことを表す．つまり $p_{ija}=1$ は $x_{ij}=a$ を，$p_{ija}=0$ は $x_{ij} \neq a$ を意味する．すると x_{ij} が 1 から 9 の値をとるという条件 (1) は，

$$p_{ij1} \vee p_{ij2} \vee \cdots \vee p_{ij9} \tag{4}$$

$$\neg p_{ija} \vee \neg p_{ijb} \qquad (1 \leq a < b \leq 9) \tag{5}$$

と書ける[☆4]．式 (4) で p_{ij1} から p_{ij9} の少なくとも 1 つが真になり，式 (5) [☆5] で p_{ij1} から p_{ij9} のうち 2 つ以上が同時に真になることを防いでいる．たとえば $\neg p_{ij1} \vee \neg p_{ij2}$ は，p_{ij1} と p_{ij2} が同時に真にならないこと，つまり x_{ij} が 1 と 2 という異なる値を同時にとらないことを意味する．

次に，条件 (2) の $x_{ij} \neq x_{kl}$ の符号化はどうすれば良いだろうか．符号化する方法の 1 つに，条件に違反する場合の否定を列挙するやり方がある．**図 -3** は，$x_{ij} \neq x_{kl}$ に違反する場合を×印で示している．たとえば $x_{ij}=1$ かつ $x_{kl}=1$ は条件に違反する．つまり $\neg(p_{ij1} \wedge p_{kl1})$ であり，これから $\neg p_{ij1} \vee \neg p_{kl1}$ が得られる[☆6]．1 だけでなくほかの値についても同様だから，この条件は，

$$\neg p_{ija} \vee \neg p_{kla} \qquad (1 \leq a \leq 9) \tag{6}$$

という論理式で表せる．

最後に残った $x_{ij}=a$ の符号化は簡単だ．単に

$$p_{ija} \tag{7}$$

で良い．

以上の (4) から (7) の式は，いくつかの**リテラル**（命題変数あるいは命題変数の否定）を OR で結ん

☆4　"∨" は OR，"¬" は NOT を表す．
☆5　36 個の式になる．
☆6　ド・モルガンの法則 $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$ を使う．

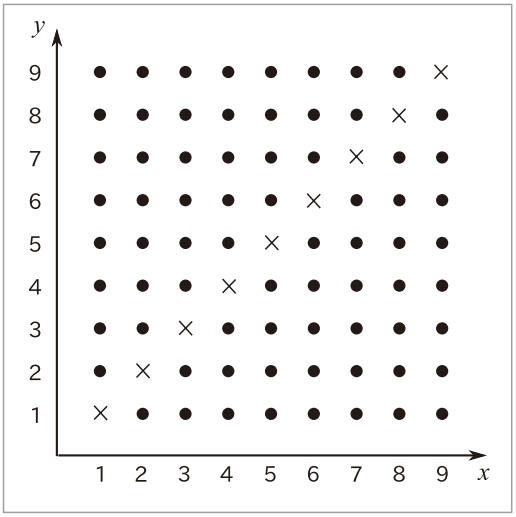


図 -3　$x \neq y$ に違反する点（× 印）

3	6	7	1	8	9	2	5	4
1	9	2	4	5	7	3	6	8
8	4	5	6	3	2	1	7	9
7	1	3	2	9	8	5	4	6
2	5	6	7	4	1	8	9	3
4	8	9	5	6	3	7	2	1
9	7	1	3	2	4	6	8	5
6	2	8	9	1	5	4	3	7
5	3	4	8	7	6	9	1	2

(a)

7	2	8	9	3	6	5	1	4
9	4	3	1	5	8	6	7	2
5	6	1	4	7	2	9	3	8
8	3	4	7	6	5	2	9	1
2	1	7	8	4	9	3	6	5
6	5	9	2	1	3	8	4	7
1	8	6	3	2	4	7	5	9
3	7	2	5	9	1	4	8	6
4	9	5	6	8	7	1	2	3

(b)

図 -4　図 -2 の数独の答え

だ形になっており，**節**と呼ばれる．SAT ソルバーには，これらの節を AND（∧）で結んだ **CNF 式**[☆7] を入力として与える（実際の入力ファイルでは 1 行ごとに 1 つの節を記述する）．

以上の符号化方法を図 -2 (a) の問題に適用すると，81×9＝729 個の命題変数，81＋81×36＋810×9＋25＝10,312 個の節からなる CNF 式になる．多いと思うかもしれないが，最新の SAT ソルバーからすると，かなり簡単な問題だ．実際，国産の優秀な SAT ソルバーである **GlueMiniSat** で試してみたところ，20 ミリ秒以内で解が求まった（**図 -4**）．

驚くべきは，答えを探すプログラムをまったく書かなかった点だ．分量は多いが，解の条件を節として羅列し，SAT ソルバーに入力として与えれば，答えが求まってしまう．SAT ソルバーは，解を探索するシステムを実現するための打出の小槌のように思える．

★SAT ソルバーはどのように動くのか

SAT ソルバーはこの問題をどのように解くのだろうか．(a) の問題を例に説明する．

例にはヒントが 25 カ所あり p_{141} 等が単位節[☆8] として存在している．一方，節 (5) には $\neg p_{141} \vee \neg p_{142}$ が含まれている．このとき，SAT ソルバーは単位伝播[☆9] と呼ばれる処理を行い $p_{142}=0$ を導く．同様に，節 (6) に対しても単位伝播の処理が行われる．たとえば $\neg p_{141} \vee \neg p_{191}$ という節から $p_{191}=0$ が導かれる．ほかの 24 個のヒントも同様に処理される．

単位伝播の処理は，まだまだ続く．5 行目 9 列目を見てみる．$p_{591}=0$ が $p_{561}=1$ から導かれる．同様に p_{592} および p_{594} から p_{599} のすべてが 0 になる．すると，節 (4) を用いた単位伝播が実行され $p_{593}=1$ が導かれる．つまり 5 行目 9 列目が 3 であることが分かった．

実は，(a) はこの単位伝播だけで解けてしまう．なんと，バックトラックの処理をまったく行わずに解が得られるのだ．

しかし，より難しい (b) の場合は，バックトラックによる試行錯誤を必要とする．**GlueMiniSat** では 41 回のバックトラックが生じた．SAT ソルバーはバックトラック時に失敗の原因を分析した結果を**学習節**として保存し，探索の枝刈りに利用している．**GlueMiniSat** が最初に獲得した学習節を見てみると，2 行目 9 列目が 9 でないことを表す $\neg p_{299}$ だった．これは仮定 p_{299} から矛盾が生じたことを意味するが，皆さんは矛盾を導けるだろうか．

★数独の SAT 符号化の工夫

単位伝播を促すようなうまい節（プロパゲーターと呼ばれることがある）を追加すると，(b) の問題

☆7 Conjunctive Normal Form．連言標準形．
☆8 1 つのリテラルからなる節のこと．
☆9 節中で，1 つを除いてすべてが偽のとき，残りの 1 つを真にする処理のこと．

でもバックトラックなしで解ける！

数独の場合，各行（あるいは各列，各ブロック）には，1 から 9 の数が必ず 1 回は現れる．たとえば，(b) の 9 列目中には 7 という数が必ず現れるはずだ．したがって，

$$p_{197} \vee p_{297} \vee p_{397} \vee \cdots \vee p_{997} \tag{8}$$

という節を加えることができる．この節があれば $p_{697}=1$ を単位伝播で導くことが可能になる．同様な節を合計で $27 \times 9=243$ 個加えると，(b) の問題もバックトラックなしで解けるようになる．

これは数独を解く手筋の 1 つを SAT ソルバーに教えているとも考えられる．数独を人手で解くときには，さまざまな手筋が知られている．ほかの手筋を節で表現できるかどうかは，読者への課題としたい．

ナンバーリンクを解いてみよう

次の題材として，ナンバーリンクを取り上げよう．上下左右に隣接するマスとの間に線を引き，同じ数字同士を交差しない線で繋ぐパズルだ．**図 -5** は問題例，**図 -6** はその答えである．

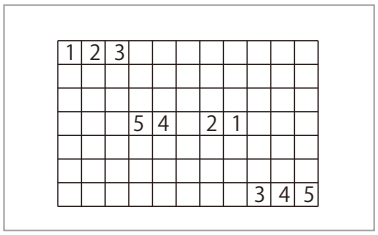


図 -5
ナンバーリンクの例

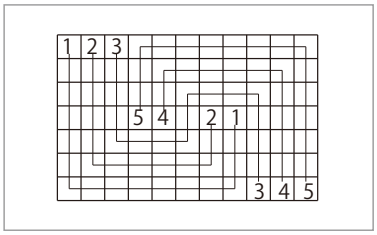


図 -6
ナンバーリンクの例の答え

★ナンバーリンクの SAT 符号化

ナンバーリンクの問題を解くプログラムを通常のプログラミング言語で書くのは難しい．しかし，SAT ソルバーを使えば簡単だ．

数独と同様に，まず整数上の条件からなる制約モデルを作ってから，その SAT 符号化を考えることにしよう．

まず，マス間を結ぶ線に関する条件から始める．下端を除く各マス (i, j) [☆10] から下のマスへの線の有無を変数 s_{ij} で表し，右端を除く各マス (i, j) から右のマスへの線の有無を変数 e_{ij} で表す．

$$s_{ij} \in \{0, 1\}, \quad e_{ij} \in \{0, 1\} \tag{9}$$

答えの図 -6 を見てみると，数が記入されている数字マスからは 1 本だけ線が出ており，空白の白マスからは 2 本線が出ていることが分かる．ただ，白マスを通らないこともあり得る．

どちらの条件も上の s_{ij} , e_{ij} を使って表せる．たとえば，数字マス (4, 4) から線が 1 本だけ出ることは，

$$s_{34} + e_{43} + s_{44} + e_{44} = 1 \tag{10}$$

と書ける．

白マス (2, 2) については 2 本または 0 本の線が出る．この条件は，次のように書ける．

$$s_{12} + e_{21} + s_{22} + e_{22} \leq 2 \tag{11}$$

$$s_{12} + e_{21} + s_{22} + e_{22} \neq 1 \tag{12}$$

すべての数字マスと白マスについて同様に記述すれば，数字マスと数字マスを交差しない線で結ぶ条件を表したことになる[☆11]．

しかし，これだけでは違う数字マス同士が結ばれる可能性がある．そこで，各マスがどの数字マスと繋がっているかを表す変数 x_{ij} を導入する．x_{ij} の取り得る値は問題に現れている数だ．例の場合は 1 から 5 になる．

$$x_{ij} \in \{1, 2, 3, 4, 5\} \tag{13}$$

当然，数字マスはその値をとる．たとえばマス (4, 4) には 5 が記入されているので以下の条件を加える．

$$x_{44}=5 \tag{14}$$

☆10 i 行目 j 列目の位置を (i, j) で表すことにする．

☆11 白マスだけを通るループの存在を許しているが，単に取り除けば良いので，気にしないことにする．

あるマス (i, j) とその隣のマス (k, l) の間に線があれば x_{ij} と x_{kl} は同じ値になる．これを「ならば」(“⇒”) を使って表すと次のように書ける☆12．

$$(s_{ij}=1) \Rightarrow (x_{ij}=x_{(i+1)\,j}) \quad (15)$$
$$(e_{ij}=1) \Rightarrow (x_{ij}=x_{i\,(j+1)}) \quad (16)$$

以上で，ナンバーリンクの制約モデルができた．次は，この制約モデルの SAT 符号化を考える．

変数 s_{ij} と e_{ij} は 0 と 1 の値しかとらないから，そのまま命題変数として使える．このとき，式 (10) などは命題変数の和に対する条件になっており，これらは**基数制約**と呼ばれる．基数制約の SAT 符号化も，さまざまな方法が考案されている．ここでは最も単純な方法を説明する．

式 (10) に相当する基数制約 $p_1+p_2+p_3+p_4=1$ の符号化は次のようになる．

$$p_1 \vee p_2 \vee p_3 \vee p_4 \quad (17)$$
$$\neg p_i \vee \neg p_j \qquad (1 \le i < j \le 4) \quad (18)$$

最初の節は p_i のうち少なくとも 1 つが 1 であること，すなわち $p_1+p_2+p_3+p_4 \ge 1$ を表している．2 番目の節は p_i のうち 2 つ以上が 1 にならないこと，つまり $p_1+p_2+p_3+p_4 \le 1$ を表している．

式 (11) に相当する基数制約 $p_1+p_2+p_3+p_4 \le 2$ も，同様に考えれば良い．つまり p_i のうち 3 つ以上が 1 になることを禁止する．

$$\neg p_i \vee \neg p_j \vee \neg p_k \quad (1 \le i < j < k \le 4) \quad (19)$$

式 (12) に相当する基数制約 $p_1+p_2+p_3+p_4 \ne 1$ は，違反する場合の否定を列挙することで以下のように表せる．

$$\neg p_1 \vee p_2 \vee p_3 \vee p_4 \quad (20)$$
$$p_1 \vee \neg p_2 \vee p_3 \vee p_4 \quad (21)$$
$$p_1 \vee p_2 \vee \neg p_3 \vee p_4 \quad (22)$$
$$p_1 \vee p_2 \vee p_3 \vee \neg p_4 \quad (23)$$

次は，式 (13) $x_{ij} \in \{1, 2, 3, 4, 5\}$ および式 (14) の $x_{ij}=a$ の符号化だ．数独の場合と同様に，命題変数 p_{ija} で x_{ij} の値が a に等しいことを表し，式 (4)(5)(7) と同じようにすれば良い．

最後は条件 (15)(16) だが，まず“⇒”の右辺の $x_{ij}=x_{kl}$ の符号化を考えよう．$x_{ij}=x_{kl}$ は，

$$(p_{ija} \Rightarrow p_{kla}) \wedge (p_{kla} \Rightarrow p_{ija}) \quad (1 \le a \le 5)$$

で良い．$A \Rightarrow B$ と $\neg A \vee B$ が同値になる事実と分配法則☆13 を使えば，以下のように書ける．

$$\neg s_{ij} \vee \neg p_{ija} \vee p_{(i+1)ja} \quad (1 \le a \le 5) \quad (24)$$
$$\neg s_{ij} \vee p_{ija} \vee \neg p_{(i+1)ja} \quad (1 \le a \le 5) \quad (25)$$
$$\neg e_{ij} \vee \neg p_{ija} \vee p_{i\,(j+1)a} \quad (1 \le a \le 5) \quad (26)$$
$$\neg e_{ij} \vee p_{ija} \vee \neg p_{i\,(j+1)a} \quad (1 \le a \le 5) \quad (27)$$

以上で，ナンバーリンクの SAT 符号化ができた．符号化の方法はやや複雑だったが，結果の CNF 式の規模は大きくない．図 -5 の例で，521 個の命題変数，2,693 個の節となる．**GlueMiniSat** で実行した所，数ミリ秒で解が求まった．

数独の場合と同様に，SAT 符号化を工夫することは可能だ．筆者らは，ここで述べた制約モデルを改良したものを用い，情報処理学会の DA シンポジウム内で行われたアルゴリズムデザインコンテスト（ナンバーリンクが題材）において 2 年連続優勝した．そこでは，種々の工夫を行い高速化を実現し，人間だと何十分もかかりそうな 35 行 48 列の問題でも 30 秒程度で解を求めることができている．

SAT 型制約ソルバーの薦め

ここまでで，SAT ソルバーを使ってパズルを解く方法を紹介した．しかし，SAT 符号化を行うのは面倒だ．制約モデルから SAT 問題へ自動で符号化ができるならそのほうが便利だろう．

そのようなシステムは SAT 型制約ソルバーと呼ばれ，さまざまなシステムが開発・公開されている．

筆者らも **Sugar**☆14, **Copris**☆15 などを公開している．

☆12 下端や右端のマスに対する場合は不要である．

☆13 $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

☆14 http://bach.istc.kobe-u.ac.jp/sugar/

☆15 http://bach.istc.kobe-u.ac.jp/copris/

```
; 変数の宣言
(int x_1_1 1 9)
  .....
(int x_9_9 1 9)
; 各行・各列・各ブロックの制約
(alldifferent x_1_1 x_1_2 x_1_3
    x_1_4 x_1_5 x_1_6 x_1_7 x_1_8 x_1_9)
  .....
(alldifferent x_9_1 x_9_2 x_9_3
    x_9_4 x_9_5 x_9_6 x_9_7 x_9_8 x_9_9)
  .....
(alldifferent x_1_1 x_1_2 x_1_3
    x_2_1 x_2_2 x_2_3 x_3_1 x_3_2 x_3_3)
  .....
; ヒント
(= x_1_3 7)
  .....
```

図 -7　Sugar で記述した数独の制約モデル（一部略）

```
import jp.kobe_u.copris._; import dsl._

val n = 9; val m = 3
for (i <- 1 to n; j <- 1 to n)
  int('x(i,j), 1, n)
for (i <- 1 to n)
  add(Alldifferent((1 to n).map('x(i,_))))
for (j <- 1 to n)
  add(Alldifferent((1 to n).map('x(_,j))))
for (i <- 1 to n by m; j <- 1 to n by m) {
  val xs =
    for (di <- 0 until m; dj <- 0 until m)
    yield 'x(i+di,j+dj)
  add(Alldifferent(xs))
}
add('x(1,3) === 7); ...; add('x(9,7) === 9)
```

図 -8　Copris で記述した数独の制約モデル

Sugar は制約モデルを入力として受け取り，符号化した結果の CNF 式を出力し，SAT ソルバーを呼び出すことで解を求める．ただし，符号化方法としては直接符号化ではなく，**順序符号化**☆16 と名付けた方法を用いている [2]．**図 -7** に Sugar で数独を解くための入力ファイル例を示す☆17．Lisp 風の表記を用い，数独の制約モデルが記述されている．

一方，Copris はプログラミング言語 Scala 上に実現された制約プログラミング用のドメイン特化言語である．Scala が提供する高度な記述力により，分かりやすく簡潔に制約モデルを記述できる．**図 -8** に Copris で数独の制約モデルを記述した例を示す．

Sugar および Copris の Web ページでは，各種のパズルソルバーを始め，さまざまな応用を紹介している．なかでもノノグラムのソルバーは，高性能と評価されているようだ．倉庫番のソルバー（Copris で記述して約 300 行）も公開しているが，現状では専用ソルバーの性能には及んでいない．読者諸氏による挑戦を期待したい．

なお，最初に紹介した Knuth の著書 [3] の中にも，ライフゲーム☆18 など楽しい応用例が紹介されている．一読をお薦めする．最後に，パズルに関する Knuth の文を紹介し，この稿を終えよう．

☆16 $x = a$ ではなく $x \geq a$（あるいは $x \leq a$）を命題変数に割り当てる方法．数式を符号化したとき，単位伝播と相性が良い．
☆17 図中の alldifferent は，与えられた引数が互いに異なることを表す．
☆18 2 次元セル・オートマトンの一種．生命の誕生・消滅を模している．

While writing this chapter, the author couldn't help relishing the fact that puzzles are now more fun than ever, as computers get faster and faster, because we keep getting more powerful dynamite to play with. —Knuth, D. E., *TAOCP*, Vol.4A, p.9.

参考文献

1) 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, pp.57-67 (2010).
2) 田村直之，丹生智也，番原睦則：制約最適化問題と SAT 符号化，人工知能学会誌，Vol.25, No.1, pp.77-85 (2010).
3) Knuth, D. E. : Satisfiability, Vol.4, Fascicle 6 of *The Art of Computer Programming,* Addison-Wesley Professional (2015).

(2016 年 4 月 29 日受付)

田村直之（正会員） tamura@kobe-u.ac.jp

神戸大学情報基盤センター教授．学術博士．論理プログラミング，制約プログラミング，SAT 技術，パズル等に興味を持つ．

宋　剛秀 soh@lion.kobe-u.ac.jp

神戸大学情報基盤センター助教．博士（情報学）．SAT 技術およびそれらの応用，制約プログラミング，システム生物学等に興味を持つ．

番原睦則（正会員） banbara@kobe-u.ac.jp

神戸大学情報基盤センター准教授．博士（工学）．制約プログラミング，SAT 技術，解集合プログラミング等に興味を持つ．