特　集

制約最適化問題とSAT符号化

Constraint Optimization Problems and SAT Encodings

田村 直之
Naoyuki Tamura
神戸大学学術情報基盤センター
Information Science and Technology Center, Kobe University
tamura@kobe-u.ac.jp

丹生 智也
Tomoya Tanjo
神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University
tanjo@stu.kobe-u.ac.jp

番原 睦則
Mutsunori Banbara
神戸大学学術情報基盤センター
Information Science and Technology Center, Kobe University
banbara@kobe-u.ac.jp

keywords: SAT encoding, constraint optimization problems, constraint satisfaction problems

1. は　じ　め　に

制約充足問題および**制約最適化問題**は，それぞれ与えられた制約を満たす解および最適解を探索する問題である．人工知能研究等で生じる多くの組合せ問題は，制約充足問題あるいは制約最適化問題として定式化できる [Barták 98, Bordeaux 06]．なお，以下では特別に区別する必要がない限り，両者をまとめて制約充足問題と呼ぶ．

一方，命題論理式の充足可能性判定問題である **SAT** は，与えられた命題論理式を真にする解を探索する問題である．

本稿では，整数上の制約充足問題を SAT 問題に変換し，SAT 問題を解く **SAT ソルバー** (SAT solver) を用いて元の問題の解を探索する **SAT 符号化** (SAT encoding) の手法について解説する [Prestwich 09]．

まず，制約充足問題の定義および問題例を紹介する．問題例としては，後述の性能評価にも使用するグラフ彩色問題を取り上げる．

次に，SAT 符号化のための基本的な手法，特に Tseitin 変換と呼ばれる方法について述べた後，各種の SAT 符号化法を紹介する．紹介する SAT 符号化法は，古くから利用されれている直接符号化法 (direct encoding)，多値符号化法 (multivalued encoding)，支持符号化法 (support encoding)，対数符号化法 (log encoding)，および最近提案された対数支持符号化法 (log-support encoding)，順序符号化法 (order encoding) である．

最後に，グラフ彩色問題を例題とした実験結果を含め，これらの各種 SAT 符号化法について比較する．

なお，SAT ソルバーのより詳しい解説については，本特集号に含まれる [井上 10, 鍋島 10b] を参照されたい．

2. 制約充足問題と制約最適化問題

2·1 制 約 充 足 問 題

制約充足問題 (Constraint Satisfaction Problem; CSP) は，各変数 (variable) に与えられたドメイン (domain) から値を割り当てることで，与えられた制約 (constraint) のすべてを満たすことができるかどうかを判定する問題である [Barták 98, Bordeaux 06]．すべての制約を満たす値割り当て (assignment) が存在する場合，元の制約充足問題は**充足可能** (satisfiable) であるといい，その値割り当てが**解**となる．値割り当てが存在しない場合，元の制約充足問題は**充足不能** (unsatisfiable) であるという．

制約充足問題は，一般には実数や集合など様々なドメイン上で展開されるが，本解説では実用上多くの応用を含む整数の有限領域 (finite domain) 上の制約充足問題を対象とする．

整数有限領域上の制約充足問題は，形式的には以下のように定義できる．

【定義 1】（制約充足問題） 制約充足問題は以下を満たす組CSP(X, Dom, C)である．

(1) X は整数変数の有限集合 $\{x_1, x_2, \ldots, x_n\}$ である．
(2) 関数 $Dom : X \to \mathcal{F}in(\mathbf{Z})$ は，各変数の取り得る値集合 (ドメイン) を定める ($\mathcal{F}in(\mathbf{Z})$ は整数 $\mathbf{Z}$ の有限部分集合の全体を表す)．すなわち，各変数 $x_i \in X$ について $Dom(x_i) \subset \mathbf{Z}$ であり，$Dom(x_i)$ は有限集合である．
(3) C は X 上の制約の有限集合 $\{C_1, C_2, \ldots, C_m\}$ であり，制約の連言を表す．

制約には，算術論理演算等で条件が記述されている内包的制約，制約を満足する (あるいは制約に違反する) 値の組の集合が陽に与えられている外延的制約，そして alldifferent 等に代表されるいわゆるグローバル制約がある．

内包的制約 (intensional constraint) は，通常の算術演算，算術比較に加え，否定 (¬)，連言 (∧)，選言 (∨)，含意 (→)，同値 (↔) 等の論理演算を用いて条件を表したものである．例えば，$(x_1+2\leq x_2)\vee(x_2+3\leq x_1)$ は内包的制約である．

外延的制約 (extensional constraint) では，制約を満足する値の組の集合である**支持点集合** (set of supports)，あるいは制約に違反する値の組の集合である**違反点集合** (set of conflicts) が与えられる．例えば，支持点集合 $R=\{(0,0),(1,1),(2,2)\}$ とし，整数変数 x_1,x_2 のドメインを $\{0,1,2\}$ とする．この時，外延的制約 $R(x_1,x_2)$ は $x_1=x_2$ を表す．同様に，$R=\{(0,0),(1,1),(2,2)\}$ が違反点集合の場合，外延的制約 $\bar{R}(x_1,x_2)$ は $x_1\neq x_2$ を表す ($\bar{R}$ は R の補集合を表す)．

グローバル制約 (global constraint) は，複数の変数に対する複雑な (しかし意味のある) 条件を簡潔に表すために導入された．例えば，alldifferent$(x_1,x_2,\ldots,x_n)$ は，x_i が互いに異なることを表す．$x_i\neq x_j$ を個別に記述するよりも簡潔になり，また効率良い解法アルゴリズムの存在が知られている [Gent 08]．このようなグローバル制約は，記述性および効率性の向上を目的として制約ソルバーや制約プログラミングシステムに数多く取り入れられている．これらのグローバル制約を集約した Global Constraint Catalog[*1]には，300 以上のグローバル制約が紹介されている．

2·2 制約最適化問題

単純に条件を満たす解を探索するだけでなく，制約を満たす解のうち最適なものを求める問題を**制約最適化問題** (Constraint Optimization Problem; COP) という．制約最適化問題では，条件を満たす解のうち，指定された目的関数 (objective function) あるいは目的変数 (objective variable) の値を最小 (あるいは最大) にする解を求めることが目的である．

整数有限領域上の制約最適化問題は，形式的には以下のように定義できる．

【定義 2】（制約最適化問題） 制約最適化問題は以下を満たす組COP(X,Dom,C,v)である．

(1) CSP(X,Dom,C) は制約充足問題である．
(2) 変数 $v\in X$ は最小化 (あるいは最大化) すべき目的変数を表す．

2·3 制約ソルバー

制約充足問題の解を探索するプログラムは，**制約ソルバー** (constraint solver) と呼ばれる．また，プログラミング言語の一部に (あるいはライブラリとして) 制約ソルバーが組込まれ，プログラミング言語とより融合したシステムは**制約プログラミング** (constraint programming) システムと呼ばれる．

*1 http://www.emn.fr/x-info/sdemasse/gccat/

8	1	6
3	5	7
4	9	2

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

図 1 3×3 の魔方陣

これらの制約ソルバーや制約プログラミングシステムには，clp(FD) [Codognet 96], ILOG Solver[*2], Choco [The choco team 08], Cream[*3]などがある．これらのシステムは，各制約に違反する値を削除する**整合性アルゴリズム** (consistency algorithm) を用いフォワード・チェック法 (forward checking) やアーク整合性維持法 (maintaining arc consistency; MAC) といった制約伝播 (constraint propagation) アルゴリズムにより探索空間を削減する工夫を行っている．また，単純な**バックトラック法** (backtracking) による探索だけでなく，矛盾の原因になった値割り当てに後戻りする**バックジャンプ法** (backjumping) などを用いて効率的な探索を実現している．これらのアルゴリズムの詳細は本稿の範囲を越えるため，文献 [Barták 98, Bordeaux 06] 等を参照されたい．

制約プログラミングシステムでは，ベースとなっている Prolog, C++, Java などの言語の構文を用いて制約充足問題を記述するが，制約充足問題を記述するための**制約モデリング言語** (constraint modeling language) を用いる場合もある．OPL [Hentenryck 99], Zinc [Banda 06], XCSP などは制約モデリング言語の例である．特に，XCSP は国際 CSP ソルバー競技会で使用され，多数のベンチマーク問題が公開されている．

2·4 制約充足問題の例

ここでは，制約充足問題の例として魔方陣とグラフ彩色問題を取り上げる．その他の問題例については，XCSP[*4]や CSPLib[*5] の Web サイトなどを参照されたい．

3×3 の魔方陣は，1 から 9 の数字を 3 行 3 列のマスに配置し，各行，各列および二つの対角線について配置されている三つの数字の和がいずれも 15 となるようにする問題である．図 1 の左は解の一つを表している．

これを制約充足問題として定式化するために，図 1 の右側のように整数変数を各マスに割り当て，各変数のドメイン $Dom(x_i)$ を $\{1,2,\ldots,9\}$ と定める．

必要な制約は，各マスの数字が互いに異なることを表す alldifferent$(x_1,x_2,\ldots,x_9)$，および各行，各列および二つの対角線の和が 15 に等しいことを表す $x_1+x_2+x_3=15$ 等である．

次の例として，最適化コンパイラでのレジスタ割り付け，無線の周波数割り当て等の応用があるグラフ彩色問

*2 http://www.ilog.com/products/cp/
*3 http://bach.istc.kobe-u.ac.jp/cream/
*4 http://www.cril.univ-artois.fr/~lecoutre/benchmarks.html
*5 http://www.csplib.org

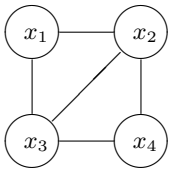

図 2 グラフ彩色問題の例

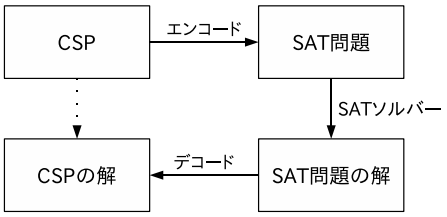

図 3 SAT 型制約ソルバー

題を取り上げる．有名なパズルである数独もグラフ彩色問題として定式化できる．

グラフ彩色問題 (Graph Coloring Problem; GCP) は，与えられた有限無向グラフ G について，隣接する頂点が同色にならないように各頂点を彩色する時，必要となる最小の色数を求める問題である．最小の色数は**彩色数** (chromatic number) と呼ばれ，$\chi(G)$ で表される．

彩色数を求める問題は NP 困難であることが知られている．また，与えられた自然数 $k \geq 3$ について，グラフが k 色以下で彩色可能かどうかを決定する問題は NP 完全である．グラフが k 色以下で彩色可能な時，そのグラフは **k-彩色可能** (k-colorable) であるという．

例えば，図 2 に示されているグラフは，3-彩色可能であるが 2-彩色可能ではない．したがって，このグラフの彩色数は 3 である．

グラフが k-彩色可能かどうかを決定する問題は，制約充足問題として定式化できる．グラフの各頂点に対して整数変数 x_i を割り当て，各整数変数のドメイン $Dom(x_i)$ を $\{0,1,\ldots,k-1\}$ と定める．また，すべての辺について，対応する整数変数の値が異なることを意味する $x_i \neq x_j$ を制約として付け加える．

例えば，図 2 に示されているグラフが 3-彩色可能かどうかを決定する問題は，制約充足問題 CSP(X, Dom, C) として以下のように定式化できる．

$$
\begin{aligned}
X &= \{x_1, x_2, x_3, x_4\} \\
Dom(x_i) &= \{0,1,2\} \qquad (i = 1,2,3,4) \\
C &= \{x_1 \neq x_2,\ x_1 \neq x_3, \\
&\qquad x_2 \neq x_3,\ x_2 \neq x_4,\ x_3 \neq x_4\}
\end{aligned}
$$

彩色数を求める問題を制約最適化問題として定式化する場合，事前に彩色数の上界を求めておく必要がある．彩色数はグラフの最大次数 $+1$ 以下であることが知られているのでこれを用いるか，貪欲法等の単純なアルゴリズムで上界を求めれば良い．

3. 制約充足問題と制約最適化問題の SAT 符号化

SAT (Boolean satisfiability testing) は，与えられた命題論理式を真にするような命題変数への真理値割り当て (truth assignment) が存在するか否かを判定する問題である．真にする真理値割り当てが存在すれば元の命題論理式は**充足可能** (satisfiable) と呼ばれ，存在しなければ**充足不能** (unsatisfiable) と呼ばれる．与えられた SAT 問題 (SAT problem instance) が充足可能な場合，その問題を真にする真理値割り当てが**解**となる．

通常，命題論理式は**連言標準形** (Conjunctive Normal Form; CNF) で与えられる．すなわち，全体の論理式はいくつかの**節** (clause) の連言 (AND) であり，各節はいくつかの**リテラル** (literal) の選言 (OR)，各リテラルは**命題変数** (propositional variable) あるいはその否定である．

SAT 型制約ソルバー (SAT-based constraint solver) は，与えられた制約充足問題を SAT 問題に符号化し，SAT ソルバーを用いて元の問題の解を求めるシステムである (図 3)．

以下では，SAT 符号化の基礎について述べた後，各種 SAT 符号化法を解説する．

3·1 SAT 符号化の基礎

任意の命題論理式は，ド・モルガン律等を用いて，否定が命題変数の直前のみに現れる**否定標準形** (Negative Normal Form; NNF) に変換できる．否定標準形の命題論理式は，分配律等を用いて連言標準形に変換できるが，分配律を用いた場合，得られる論理式の長さが元の論理式の長さの指数関数的になる可能性がある．

Tseitin は，元の論理式の長さと線形の長さの連言標準形に変換する **Tseitin 変換** (Tseitin transformation) の方法を示した [Prestwich 09]．Tseitin 変換では，$P \vee (Q \wedge R)$ 等について，新しい命題変数 p を導入し，$Q \wedge R$ の部分を p で置き換えるとともに，$p \leftrightarrow (Q \wedge R)$ と同値な論理式として $(\neg p \vee Q) \wedge (\neg p \vee R) \wedge (p \vee \neg Q \vee \neg R)$ を追加する．ただし，元の論理式が否定標準形であれば，以下で示すように $(\neg p \vee Q) \wedge (\neg p \vee R)$ のみの追加で良い．

Tseitin 変換を行う前の論理式と変換後の論理式は，同値ではないが，両者の充足可能性が一致する**充足同値** (equi-satisfiable) になっている．

［定理 1］（Tseitin 変換） A を否定標準形の命題論理式, B を命題論理式とし，命題変数 p は B 中に現れておらず A 中には $\neg p$ の形で現れていないとする．$A[B]$ により，A 中のすべての p の出現を B で置き換えた論理式を表す時，$A[B]$ と $A \wedge (\neg p \vee B)$ は充足同値であり，以下が成り立つ．

$$A[B] \text{ が充足可能} \iff A \wedge (\neg p \vee B) \text{ が充足可能}$$

(証明) ($\Longrightarrow$) $A[B]$ を充足する真理値割り当てを α とする．$A[B]$ 中に p が現れないことから $\alpha'(p)=\alpha(B)$, $\alpha'(q)=\alpha(q)$ ($q\neq p$ の時) により α' を定める．明らかに $\alpha'(A)=\alpha(A[B])=1$ であり，$\alpha'(p)=\alpha(B)=\alpha'(B)$ より $\alpha'(\neg p\vee B)=1$ となり，α' は $A\wedge(\neg p\vee B)$ を充足する．
($\Longleftarrow$) $A\wedge(\neg p\vee B)$ を充足する真理値割り当てを α' とすると，$\alpha'(A)=1$ かつ $\alpha'(\neg p\vee B)=1$ である．α' の定義域から p を削除したもので α を定める．この時，$\neg p$ が現れていない任意の否定標準形の論理式 C について，$\alpha'(C)=1$ ならば $\alpha(C[B])=1$ であることを，C に関する構造的帰納法で証明する．

C が命題変数 p の場合，$\alpha(C[B])=\alpha(B)=\alpha'(B)$ だが，$\alpha'(p)=1$ と $\alpha'(\neg p\vee B)=1$ より $\alpha'(B)=1$ である．

C が命題変数 q あるいは $\neg q$ (ただし q は p と異なる) の場合，$\alpha(C[B])=\alpha(C)=\alpha'(C)=1$ である．

C が $C_1\wedge C_2$ の場合，$\alpha'(C)=1$ より $\alpha'(C_1)=1$ かつ $\alpha'(C_2)=1$ である．帰納法の仮定より $\alpha(C_1[B])=1$ かつ $\alpha(C_2[B])=1$，したがって $\alpha(C[B])=1$ である．

C が $C_1\vee C_2$ の場合，$C_1\wedge C_2$ の場合と同様にして $\alpha(C[B])=1$ が示される．

よって，α は $A[B]$ を充足する． □

例えば，否定標準形の論理式 $P\vee(Q\wedge R)$ について Tseitin 変換を行う場合，新しい命題変数 p を準備し，A を $P\vee p$, B を $Q\wedge R$ とする．上の定理より $P\vee(Q\wedge R)$ と $(P\vee p)\wedge(\neg p\vee(Q\wedge R))$ すなわち $(P\vee p)\wedge(\neg p\vee Q)\wedge(\neg p\vee R)$ は充足同値である．このような変換を，同様の部分論理式に対して繰り返し行えば，最終的に連言標準形の論理式が得られる．

3·2 SAT 符号化法の分類

後述のように制約充足問題の SAT 符号化法には様々な方法が存在する [Prestwich 09]．これらは，以下の 2 種類の観点から分類できる．

整数変数の符号化法: 各整数変数 x をどのように命題変数で表現し，どのような節に符号化するか．

制約の符号化法: 各制約をどのような節に符号化するか．

整数変数の符号化法については，以下の 3 種類が知られている．

(1) 各整数変数 x と各整数定数 $a\in Dom(x)$ に対して，$x=a$ を意味する命題変数 $p(x=a)$ を用いる方法．

(2) 各整数変数 x の 2 進表現に着目し，x の i 桁目が 1 に等しいことを意味する命題変数 $p(x^{(i)})$ を用いる方法．

(3) 各整数変数 x と各整数定数 $a\in Dom(x)$ に対して，$x\leq a$ を意味する命題変数 $p(x\leq a)$ を用いる方法．

(1) は直接符号化法，多値符号化法，支持符号化法で，(2) は対数符号化法，対数支持符号化法で，(3) は順序符号化法で用いられている方法である．

3·3 直接符号化法と多値符号化法

直接符号化法 (direct encoding) は最も広く用いられている SAT 符号化法である．この符号化法では，各整数変数 x と各整数定数 $a\in Dom(x)$ に対して，$x=a$ を意味する命題変数 $p(x=a)$ を用いる [de Kleer 89, Walsh 00]．

各整数変数 x がそのドメイン $\{a_1,a_2,\ldots,a_n\}$ 中の値を少なくとも一つ取るという条件は，以下の **at-least-one 節**で表される．

$$p(x=a_1)\vee p(x=a_2)\vee\cdots\vee p(x=a_n)$$

また，ドメイン中の値を二つ以上取らないという条件は，以下の **at-most-one 節**で表される．

$$\neg p(x=a_i)\vee\neg p(x=a_j)\quad(1\leq i<j\leq n)$$

例えば，変数 x のドメインが $\{0,1,2\}$ の場合，at-least-one 節と at-most-one 節は以下のようになる．

$$\begin{aligned}&p(x=0)\vee p(x=1)\vee p(x=2)\\&\neg p(x=0)\vee\neg p(x=1)\\&\neg p(x=0)\vee\neg p(x=2)\\&\neg p(x=1)\vee\neg p(x=2)\end{aligned}$$

制約については，それを違反点集合として表現し，各要素を節に符号化する．すなわち，違反点集合 R で表された制約 $\bar{R}(x_1,x_2,\ldots,x_n)$ を符号化する場合，各要素 $(a_1,a_2,\ldots,a_n)\in R$ について，以下の節を追加する．

$$\neg p(x_1=a_1)\vee\neg p(x_2=a_2)\vee\cdots\vee\neg p(x_n=a_n)$$

例えば，変数 x,y のドメインが $\{0,1,2\}$ の場合，制約 $x\neq y$ は以下のように符号化される．

$$\begin{aligned}&\neg p(x=0)\vee\neg p(y=0)\\&\neg p(x=1)\vee\neg p(y=1)\\&\neg p(x=2)\vee\neg p(y=2)\end{aligned}$$

直接符号化法の改良として**多値符号化法** (multivalued encoding) がある [Selman 92]．多値符号化法では，ドメイン・サイズの 2 乗個ある at-most-one 節を省略する．これは，制約を違反点集合として表現する場合，各整数変数がただ一つの値を取るという条件が不要なためである．この時，整数変数 x について，複数の $p(x=a)$ が真となる場合が生じるが，いずれの値を選んでも元の制約充足問題の解となる．

他に，at-most-one 節の個数を減らす方法としては，ラダー符号化法 (ladder encoding) [Gent 04]，ビットワイズ符号化法 (bitwise encoding) [Prestwich 07] がある．

3·4 支持符号化法

支持符号化法 (support encoding) は，直接符号化法と同様に各整数変数 x と各整数定数 $a\in Dom(x)$ に対して，$x=a$ を意味する命題変数 $p(x=a)$ を用いる [Kasif 90, Gent 02]．at-least-one 節および at-most-one 節の利用も直接符号化法と同様である．

直接符号化法では制約の違反点集合に着目したが，支持符号化法では制約の支持点集合に着目する．なお，制約については 2 変数間の制約のみを対象とする．

今，2 変数制約 $R(x,y)$ に対して，$x=a$ の時に $R(x,y)$ を満たす y の値が $b_1, b_2, \ldots, b_n$ とする．この時，以下の節を追加する．

$$\neg p(x=a) \vee p(y=b_1) \vee \cdots \vee p(y=b_n)$$

この節は，$x=a \rightarrow (y=b_1 \vee \cdots \vee y=b_n)$，すなわち $x=a$ ならば y が $b_1, b_2, \ldots, b_n$ のいずれかに等しいことを表している．

同様に，$y=b$ の時に $R(x,y)$ を満たす x の値が $a_1, a_2, \ldots, a_n$ ならば，以下の節を追加する．

$$\neg p(y=b) \vee p(x=a_1) \vee \cdots \vee p(x=a_n)$$

例えば，変数 x, y のドメインが $\{0,1,2\}$ の場合，制約 $x \neq y$ は以下のように符号化される．

$$\begin{array}{l}
\neg p(x=0) \vee p(y=1) \vee p(y=2) \\
\neg p(x=1) \vee p(y=0) \vee p(y=2) \\
\neg p(x=2) \vee p(y=0) \vee p(y=1) \\
\neg p(y=0) \vee p(x=1) \vee p(x=2) \\
\neg p(y=1) \vee p(x=0) \vee p(x=2) \\
\neg p(y=2) \vee p(x=0) \vee p(x=1)
\end{array}$$

簡単に確認できるように，制約を支持符号化した節は直接符号化した節から融合 (resolution) により導出できる．また，逆も導出可能である．したがって，両者は論理的に同値であるが，後述の実験結果でも示されるように SAT ソルバーにおける性能は異なってくる．

支持符号化法を多変数制約に拡張したものとしては，[Bessière 03] がある．

3·5 対数符号化法

対数符号化法 (log encoding) は，各整数変数 x の 2 進表現に着目し，x の i 桁目 (LSB を 1 桁目とする) が 1 に等しいことを表す命題変数 $p(x^{(i)})$ を用いる [Iwama 94, Gelder 08].

なお，変数のドメインの下限は 0 以上とし，ドメインに含まれない値は，それらを除外する節を追加する．

例えば，変数 x のドメインが $\{0,1,2\}$ の場合，二つの命題変数 $p(x^{(1)}), p(x^{(2)})$ を用い，$x \neq 3$ を意味する以下の節を追加する．

$$\neg p(x^{(2)}) \vee \neg p(x^{(1)})$$

制約については，直接符号化法と同様に制約を違反点集合として表現し，違反点集合中の各組を節に符号化する．すなわち，制約に違反するすべての $x_1=a_1, x_2=a_2, \ldots, x_n=a_n$ について，$x_1 \neq a_1 \vee x_2 \neq a_2 \vee \cdots \vee x_n \neq a_n$ に相当する条件を 2 進表現で考え，それを符号化する．

例えば，変数 x, y についてドメインが $\{0,1,2\}$ の場合，制約 $x \neq y$ は，$x \neq 0 \vee y \neq 0, x \neq 1 \vee y \neq 1, x \neq 2 \vee y \neq 2$ と表せるので，それらの 2 進表現に対応した以下の節に符号化される．

$$\begin{array}{l}
p(x^{(2)}) \vee p(x^{(1)}) \vee p(y^{(2)}) \vee p(y^{(1)}) \\
p(x^{(2)}) \vee \neg p(x^{(1)}) \vee p(y^{(2)}) \vee \neg p(y^{(1)}) \\
\neg p(x^{(2)}) \vee p(x^{(1)}) \vee \neg p(y^{(2)}) \vee p(y^{(1)})
\end{array}$$

算術演算や算術比較を用いた内包的制約については，通常の論理回路と同様に論理式で演算を構成する方法も用いられる [Huang 08]．また，制約 $x \neq y$ については，x と y の対応するビットの排他的論理和を考えて符号化する方法もある [Gelder 08].

3·6 対数支持符号化法

対数支持符号化法 (log-support encoding) は，対数符号化法と同様に各整数変数 x の 2 進表現について，x の i 桁目に対応する命題変数 $p(x^{(i)})$ を用いる [Gavanelli 07]. ドメインに含まれない値に対して，それらを除外する節を追加する点も同様である．

制約については，支持符号化法と同様に支持点集合に着目して符号化する．2 変数制約のみを対象とする点も同様である．

今，2 変数制約 $R(x,y)$ に対して，$x=a$ の時に $R(x,y)$ を満たす y の値が $b_1, b_2, \ldots, b_n$ とする．ここで，支持符号化法と同様に $x=a \rightarrow (y=b_1 \vee \cdots \vee y=b_n)$ の符号化を考えるが，2 進表現を用いた場合，$y=b_i$ の部分が複数リテラルの連言となり，一つの節 (複数リテラルの選言) として表現できない場合が生じる．

そこで [Gavanelli 07] では，$y=b_1 \vee \cdots \vee y=b_n$ を複数リテラルの選言にまとめることが可能な場合のみ，上記の方法を用いて符号化する方法を提案している．複数リテラルの選言で表せない場合には，対数符号化法と同様に違反点による節に符号化する．

例えば，変数 x, y についてドメインが $\{0,1,2\}$ の場合，制約 $x \neq y$ は以下のように符号化される．

$$\begin{array}{l}
p(x^{(2)}) \vee p(x^{(1)}) \vee p(y^{(2)}) \vee p(y^{(1)}) \\
p(x^{(2)}) \vee \neg p(x^{(1)}) \vee \neg p(y^{(1)}) \\
p(y^{(2)}) \vee \neg p(y^{(1)}) \vee \neg p(x^{(1)}) \\
\neg p(x^{(2)}) \vee p(x^{(1)}) \vee \neg p(y^{(2)}) \\
\neg p(y^{(2)}) \vee p(y^{(1)}) \vee \neg p(x^{(2)})
\end{array}$$

上記はそれぞれ $\neg(x=0 \wedge y=0), x=1 \rightarrow (y=0 \vee y=2), y=1 \rightarrow (x=0 \vee x=2), x=2 \rightarrow (y=0 \vee y=1), y=2 \rightarrow (x=0 \vee x=1)$ に相当し，$x=0, y=0$ の場合のみ違反点による符号化となっている．

なお [Gavanelli 07] では，通常の 2 進表現ではなくグレイコードによる表現を用いることで，複数リテラルの選言にまとめる場合を増やす工夫も提案している．

3·7 順序符号化法

順序符号化法 (order encoding) は，各整数変数 x と各整数定数 $a \in Dom(x)$ に対して，$x \leq a$ を意味する命題変数 $p(x \leq a)$ を用いる [Tamura 06, Tamura 09].

この符号化法は，Crawford と Baker によりジョブショップ・スケジューリング問題に適用された方法 [Crawford 94, Inoue 06, Nabeshima 06] を制約充足問題に適用可能なように一般化したものである．ショップ・スケジューリング問題および 2 次元ストリップパッキング問題で未知だった最適値決定に成功する等，有望な手法であることが示されている [田村 07, 多賀 07, 越村 09, Soh 09].

順序符号化法では，各整数変数 x について，そのドメインが $\{a_1, a_2, \ldots, a_n\}$ の時 (ただし $a_1 < a_2 < \ldots < a_n$)，$x \le a_i$ を表す $n-1$ 個の命題変数 $p(x \le a_1), p(x \le a_2), \ldots, p(x \le a_{n-1})$ を用いる．なお，$x \le a_n$ は常に真であるため，命題変数 $p(x \le a_n)$ は不要である．また，これらの命題変数間の関係を表す以下の節を用いる．

$$\neg p(x \le a_i) \vee p(x \le a_{i+1}) \quad (1 \le i \le n-2)$$

例えば，変数 x のドメインが $\{0,1,2\}$ の場合，二つの命題変数 $p(x \le 0), p(x \le 1)$ を用い，以下の節を追加する．

$$\neg p(x \le 0) \vee p(x \le 1)$$

この時，上記の節を充足可能にする真理値割り当ては 3 通りあり，それぞれ $x=0, x=1, x=2$ に対応する．

$p(x \le 0)$	$p(x \le 1)$	解釈
1	1	$x=0$
0	1	$x=1$
0	0	$x=2$

制約については，制約に違反する点ではなく違反する範囲を符号化する．すなわち，範囲 $a_1 < x_1 \le b_1$，$\ldots$，$a_n < x_n \le b_n$ 中のすべての点 $(x_1, \ldots, x_n)$ が制約に違反する時，以下の節を追加する．

$$p(x_1 \le a_1) \vee \neg p(x_1 \le b_1) \vee \cdots \vee p(x_n \le a_n) \vee \neg p(x_n \le b_n)$$

線形式を用いた線形制約については，より簡潔な符号化が可能である [Tamura 09]．今，a_i を非零の整数定数，c を整数定数，x_i を互いに異なる整数変数とする．この時，制約 $\sum_{i=1}^{n} a_i x_i \le c$ は以下のように符号化できる．

$$\bigwedge_{b_i} \bigvee_{i} (a_i x_i \le b_i)^{\#}$$

ここで b_i は，$\sum_{i=1}^{n} b_i = c - n + 1$ を満たすように動くとし，変換 $()^{\#}$ は以下のように定義する．

$$(a\,x \le b)^{\#} \equiv \begin{cases} p(x \le \lfloor b/a \rfloor) & (a > 0) \\ \neg p(x \le \lceil b/a \rceil - 1) & (a < 0) \end{cases}$$

ただし，$Dom(x)$ の最小値未満の a については $p(x \le a)$ を偽に変換し，最大値以上については真に変換する．

例えば，整数変数 x, y のドメインが $\{0,1,2\}$ の時，制約 $x - y \le -1$ は以下の三つの節に符号化される．

$$\neg p(y \le 0)$$
$$p(x \le 0) \vee \neg p(y \le 1)$$
$$p(x \le 1)$$

```
(int x1 1 9)
(int x2 1 9)
(int x3 1 9)
(int x4 1 9)
(int x5 1 9)
(int x6 1 9)
(int x7 1 9)
(int x8 1 9)
(int x9 1 9)
(alldifferent x1 x2 x3 x4 x5 x6 x7 x8 x9)
(= (+ x1 x2 x3) 15)
(= (+ x4 x5 x6) 15)
(= (+ x7 x8 x9) 15)
(= (+ x1 x4 x7) 15)
(= (+ x2 x5 x8) 15)
(= (+ x3 x6 x9) 15)
(= (+ x1 x5 x9) 15)
(= (+ x3 x5 x7) 15)
```

図 4 3×3 魔方陣の Sugar での記述例

ここで $p(x \le 0) \vee \neg p(y \le 1)$ は,「$x \le 0$ または $y > 1$」であること，すなわち「$x \ge 1$ かつ $y \le 1$」が制約に違反する領域であることを表している．

$x \ne y$ 等の場合，一旦 $x - y \le -1 \vee y - x \le -1$ のように線形制約に置き換え，さらに Tseitin 変換を用いて $(p \vee q) \wedge (\neg p \vee x - y \le -1) \wedge (\neg q \vee y - x \le -1)$ と変換してから符号化する (p, q は新しい命題変数)．したがって $x \ne y$ は以下の 7 個の節に符号化される．

$$p \vee q$$
$$\neg p \vee \neg p(y \le 0)$$
$$\neg p \vee p(x \le 0) \vee \neg p(y \le 1)$$
$$\neg p \vee p(x \le 1)$$
$$\neg q \vee \neg p(x \le 0)$$
$$\neg q \vee p(y \le 0) \vee \neg p(x \le 1)$$
$$\neg q \vee p(y \le 1)$$

ただし，この場合 p と q が排他的であるので，q を $\neg p$ で置き換え，最初の節を省略することも可能である．

順序符号化法に基づいた SAT 型制約ソルバーとしては，Sugar[*6] がある [Tamura 08a, Tamura 08b, Tanjo 08]．符号化を行う部分は Java で記述されており，SAT ソルバーとしては MiniSat や PicoSAT 等が利用できる．制約充足問題だけでなく制約最適化問題や最大制約充足問題 (Max-CSP) にも対応している．

図 4 は，3×3 魔方陣の制約充足問題を Sugar で記述した例である．この例に対して Sugar を起動すると，順序符号化法により SAT 問題が作成され，その SAT 問題に対し SAT ソルバーが起動される．SAT 問題の解が得られれば，Sugar はそれを元の制約充足問題の解に逆変換し結果を表示する．

3·8 制約最適化問題の SAT 符号化

制約最適化問題COP(X, Dom, C, v)については，目的変数 v の取り得る値の上限 u (最小化問題の場合)，下限 l

*6 http://bach.istc.kobe-u.ac.jp/sugar/

(最大化問題の場合) あるいは双方を変化させながら，繰り返し制約充足問題CSP($X, Dom, C \cup \{l \leq v, v \leq u\}$)を解くことで，元の制約最適化問題の最適解を求めることができる．この時，目的変数の範囲は二分探索法に基づいて変化させるのが効率的である [Inoue 06].

しかし，この方法では複数回の SAT 符号化および SAT ソルバーの起動が必要となる．これを改善する方法として，SAT ソルバーのインクリメンタル探索 (incremental search) [Eén 03b] を用い，SAT ソルバーの連続動作および学習節の再利用を実現し最適値探索の速度を大幅に向上させることが可能である [Nabeshima 06, Tanjo 08].

4. 各種 SAT 符号化法の比較

本節では，各種 SAT 符号化法について，使用する命題変数の個数や作成される節数に関する最悪領域計算量および SAT ソルバーの単位伝播との関連について述べる．その後，グラフ彩色問題をベンチマークとして，各種 SAT 符号化法の性能評価を行う．

以下では，整数変数のドメイン・サイズを d とする．また，2 変数制約のみについて考え，内包的制約については $x \leq y$ を例とする．

直接符号化法は，各整数変数の符号化に $O(d)$ 個の命題変数を用い，at-least-one 節および at-most-one 節で合計 $O(d^2)$ 個の節を必要とする．直接符号化法の変種である多値符号化法は，at-most-one 節を省略しており at-least-one 節一つだけで良い．いずれの符号化法も，各 2 変数制約の符号化には $O(d^2)$ 個の節を必要とする．

支持符号化法での整数変数の符号化は直接符号化法と同様である．各 2 変数制約の符号化には $O(d)$ 個の節を必要とする．

対数および対数支持符号化法は，各整数変数の符号化に $O(\log d)$ 個の命題変数を用い，取り得ない値を除外するために $O(\log d)$ 個の節を必要とする [Gelder 08]．各外延的 2 変数制約の符号化には $O(d^2)$ 個の節を必要とする．内包的制約 $x \leq y$ の場合，比較を行う論理回路を構成することにより各制約を $O(\log d)$ 個の節で符号化できる [Huang 08].

順序符号化法は，各整数変数の符号化に $O(d)$ 個の命題変数を用い，命題変数間の関係を表すため $O(d)$ 個の節を必要とする．各外延的 2 変数制約には $O(d^2)$ 個の節を必要とする．内包的制約 $x \leq y$ の場合，各制約を $O(d)$ 個の節で符号化できる [Tamura 09].

SAT ソルバーの**単位伝播** (unit propagation) との関連にも注意する必要がある．単位伝播は，系統的 SAT ソルバーの基礎となっている **DPLL アルゴリズム**での基本的動作の一つである [井上 10]．以下では，DPLL アルゴリズムの利用を前提として説明を行う．

直接符号化法の場合，符号化後の SAT 問題に対して単位伝播を行う処理は，元の制約充足問題に対するフォワード・チェック法に対応する [Walsh 00]．フォワード・チェック法は，最近に具体化された変数とまだ具体化されていない変数間についてのみアーク整合性 (arc consistency) アルゴリズムを適用する方法である [Barták 98].

支持符号化法の場合，単位伝播による処理はアーク整合性維持法 (MAC) に対応する [Gent 02]．MAC はすべての変数間にアーク整合性アルゴリズムを適用する方法であり，フォワード・チェック法よりも強い制約伝播アルゴリズムとなっている [Barták 98].

対数符号化法での単位伝播は，フォワード・チェック法よりもさらに弱い制約伝播となる [Walsh 00].

順序符号化法の場合，単位伝播は元の制約充足問題における範囲伝播 (bounds propagation) に対応する．これは，整数変数の取り得る値の範囲を伝播する方法である．MAC よりも弱いが，高速な伝播が可能であるため多くの制約ソルバーで使用されている [Choi 06].

このように各符号化法には様々な差異があるが，どの符号化法を選ぶべきかは簡単な問題ではない．解くべき問題の性質および利用する SAT ソルバーの特性 (系統的ソルバーか確率的ソルバーか) によっても，異なってくる [Prestwich 09, Prestwich 03].

また，必ずしも節数やリテラル数を減らせば良いのではない．例えば，問題に対称性除去の条件を追加することで，大幅に性能が向上する場合がある．命題変数の個数についても同様である．冗長だが適切な命題変数を用いれば，矛盾からの節学習の際にその後の探索空間の枝刈りに有効な節が生成される可能性がある．

4·1 グラフ彩色問題での比較

ここでは，グラフ彩色問題を使用した実験結果を通じて，各種 SAT 符号化法の性能比較を行う．なおグラフ彩色問題の SAT 符号化については，文献 [Prestwich 03, Gelder 08, Tamura 09] などの研究がある．

グラフ彩色問題のベンチマーク問題としては，Graph Coloring and its Generalization[*7]で公開されている全 119 問のうち，彩色数 (必要となる最小の色数 $\chi(G)$) が既知[*8]の 52 問を用いた．これらの問題の頂点数は 11∼1085, 辺数は 20∼121275，彩色数は 4∼63 である．

実験は，本解説で述べた直接, 多値, 支持, 対数, 対数支持, 順序の 6 種類の符号化法を用い，色数 $k = \chi(G)$ の場合と $k = \chi(G) - 1$ の場合の 2 通りで SAT 問題に符号化した 624 問について，それぞれを制限時間 30 分として SAT ソルバーで解いた時の解けた問題数および SAT ソルバーの使用した CPU 時間を計測する方法で行った．

SAT ソルバーとしては，優れた性能で知られている MiniSat 2.0 [Eén 03a] を使用し，Xeon 3GHz，メモリー 16GB の計算機上で実行した．

*7 http://mat.gsia.cmu.edu/COLOR04/
*8 各種符号化による予備実験で彩色数を決定したものを含む．

表 1 グラフ彩色問題の解けた問題数と平均 CPU 時間

	$k=\chi(G)$		$k=\chi(G)-1$	
SAT 符号化法	解けた問題数	平均 CPU 時間 (秒)	解けた問題数	平均 CPU 時間 (秒)
直接	46	0.07	42	13.61
多値	46	0.03	42	15.45
支持	46	1.77	39	63.74
対数	46	10.40	44	2.20
対数支持	46	4.40	44	2.04
順序	46	0.96	45	2.61

表 2 グラフ彩色問題 6 問の CPU 時間 (秒) の比較 ($k=\chi(G)-1$)

グラフ彩色問題名	k	直接	多値	支持	対数	対数支持	順序
le450_15b	14	-	-	-	-	-	962.25
school1	13	-	-	-	1081.91	1081.91	535.15
school1_nsh	13	-	-	-	1541.99	1541.99	119.41
DSJR500.1	11	1770.00	858.71	-	7.13	7.13	2.05
queen8_12	11	1097.94	228.16	-	20.05	20.05	2.79
5-FullIns_4	8	642.10	670.70	-	305.51	305.51	27.89

表中の “-” は 1800 秒以内で解けなかったことを表す.

表 1 にベンチマーク問題 52 問に対する集計結果を示す．平均 CPU 時間は，すべての符号化法で解けた問題に対する値である ($k=\chi(G)$ の時 46 問，$k=\chi(G)-1$ の時 39 問).

色数 k が彩色数 $\chi(G)$ に一致する場合，6 種類のいずれの符号化法も 30 分以内に同一の 46 問について解を得た．平均 CPU 時間で見ると，多値と直接符号化法が速く，順序符号化法も比較的良い性能を示している．

$k=\chi(G)-1$，すなわち彩色不能な場合，順序符号化法が 45 問について充足不能を示し，他のどの符号化法方法よりも多くの問題を解いた．45 問中，他の符号化法で解けない場合があった 6 問の CPU 時間を表 2 に示す.

以上から，グラフ彩色問題について，順序符号化法が彩色可能な場合も彩色不能な場合も平均的に良い性能を示すといえる．

5. おわりに

本稿では，整数上の制約充足問題および制約最適化問題を SAT 問題に変換し，SAT ソルバーを用いて求解を行う SAT 符号化の手法について解説した．

このような SAT 型制約ソルバーが，通常の制約ソルバーよりも優れた性能を示す場合も報告されている．特に，順序符号化法に基づいた Sugar [Tamura 08a, Tamura 08b, Tanjo 08] は，2008 年 CSP ソルバー競技会[*9] のグローバル制約部門で第 1 位，2008 年 Max-CSP ソルバー競技会の 3 部門で第 1 位，2009 年 CSP ソルバー競技会[*10] の 3 部門で第 1 位となっている．

また，プランニングやスケジューリング問題および有界モデル検査等のシステム検証の分野でも SAT 型システムの有効性が示されている [鍋島 10a, 番原 10].

このように SAT 型システムが広がりを見せつつある理由には，近年の SAT ソルバーの優れた実装技術による高速な性能が背景にある．また，SAT ソルバーで使用されている技術が，これまでの制約ソルバー等とは異なった探索を実現している点も特徴となっている [Bordeaux 06].

しかし，本文でも述べたように，どのような SAT 符号化法を用いるべきかは大きな課題として残っている．不適切な SAT 符号化法を用いた場合，必ずしも良い結果を望めない点に注意する必要がある．

本稿での各種 SAT 符号化法の解説が，読者の研究の発展の一助となれば幸いである．

◇ 参 考 文 献 ◇

[番原 10] 番原 睦則, 田村 直之：SAT によるシステム検証, 人工知能学会誌, Vol. 25, No. 1 (2010)

[Banda 06] Banda, de la M. J. G., Marriott, K., Rafeh, R., and Wallace, M.: The Modelling Language Zinc, in *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP 2006)*, pp. 700–705 (2006)

[Barták 98] Barták, R.: On-line Guide to Constraint Programming (1998), http://kti.ms.mff.cuni.cz/~bartak/constraints/

[Bessière 03] Bessière, C., Hebrard, E., and Walsh, T.: Local Consistencies in SAT, in *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003), LNCS 2919*, pp. 299–314 (2003)

[Bordeaux 06] Bordeaux, L., Hamadi, Y., and Zhang, L.: Propositional Satisfiability and Constraint Programming: A Comparative Survey, *ACM Computing Surveys*, Vol. 38, No. 4 (2006)

[Choi 06] Choi, C. W., Harvey, W., Lee, J. H. M., and Stuckey, P. J.: Finite Domain Bounds Consistency Revisited, in *Proceedings of the 19th Australian Joint Conference on Artificial Intelligence, LNCS 4304*, pp. 49–58 (2006)

[Codognet 96] Codognet, P. and Diaz, D.: Compiling Constraints in clp(FD), *Journal of Logic Programming*, Vol. 27, No. 3, pp. 185–226 (1996)

[Crawford 94] Crawford, J. M. and Baker, A. B.: Experimental Results on the Application of Satisfiability Algorithms to Scheduling

*9 http://www.cril.univ-artois.fr/CPAI08/
*10 http://www.cril.univ-artois.fr/CPAI09/

Problems, in *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94)*, pp. 1092–1097 (1994)
[de Kleer 89] de Kleer, J.: A Comparison of ATMS and CSP Techniques, in *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 89)*, pp. 290–296 (1989)
[Eén 03a] Eén, N. and Sörensson, N.: An Extensible SAT-solver, in *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003)*, pp. 502–518 (2003)
[Eén 03b] Eén, N. and Sörensson, N.: Temporal Induction by Incremental SAT Solving, *Electronic Notes in Theoretical Computer Science*, Vol. 89, No. 4 (2003)
[Gavanelli 07] Gavanelli, M.: The Log-Support Encoding of CSP into SAT, in *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP 2007), LNCS 4741*, pp. 815–822 (2007)
[Gelder 08] Gelder, A. V.: Another Look at Graph Coloring via Propositional Satisfiability, *Discrete Applied Mathematics*, Vol. 156, No. 2, pp. 230–243 (2008)
[Gent 02] Gent, I. P.: Arc Consistency in SAT, in *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI 2002)*, pp. 121–125 (2002)
[Gent 04] Gent, I. P. and Nightingale, P.: A New Encoding of Alldifferent into SAT, in *Proceedings of the Third International Workshop on Modelling and Reformulating Constraint Satisfaction Problems* (2004)
[Gent 08] Gent, I. P., Miguel, I., and Nightingale, P.: Generalised Arc Consistency for the AllDifferent Constraint: An Empirical Survey, *Artificial Intelligence*, Vol. 172, No. 18, pp. 1973–2000 (2008)
[Hentenryck 99] Hentenryck, van P.: *The OPL Optimization Programming Language*, MIT Press (1999)
[Huang 08] Huang, J.: Universal Booleanization of Constraint Models, in *Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming (CP 2008), LNCS 5202*, pp. 144–158 (2008)
[Inoue 06] Inoue, K., Soh, T., Ueda, S., Sasaura, Y., Banbara, M., and Tamura, N.: A Competitive and Cooperative Approach to Propositional Satisfiability, *Discrete Applied Mathematics*, Vol. 154, No. 16, pp. 2291–2306 (2006)
[井上 10] 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)
[Iwama 94] Iwama, K. and Miyazaki, S.: SAT-Variable Complexity of Hard Combinatorial Problems, in *Proceedings of the IFIP 13th World Computer Congress*, pp. 253–258 (1994)
[Kasif 90] Kasif, S.: On the Parallel Complexity of Discrete Relaxation in Constraint Satisfaction Networks, *Artificial Intelligence*, Vol. 45, No. 3, pp. 275–286 (1990)
[越村 09] 越村 三幸, 鍋島 英知, 藤田 博, 長谷川 隆三：SAT 変換による未解決ジョブショップスケジューリング問題への挑戦, スケジューリング・シンポジウム 2009 講演論文集, pp. 209–213 (2009)
[Nabeshima 06] Nabeshima, H., Soh, T., Inoue, K., and Iwanuma, K.: Lemma Reusing for SAT based Planning and Scheduling, in *Proceedings of the International Conference on Automated Planning and Scheduling 2006 (ICAPS'06)*, pp. 103–112 (2006)
[鍋島 10a] 鍋島 英知：SAT によるプランニングとスケジューリング, 人工知能学会誌, Vol. 25, No. 1 (2010)
[鍋島 10b] 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)
[Prestwich 03] Prestwich, S. D.: Local Search on SAT-encoded Colouring Problem, in *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003), LNCS 2919*, pp. 105–119 (2003)
[Prestwich 07] Prestwich, S. D.: *Trends in Constraint Programming*, chapter 15: Finding Large Cliques Using SAT Local Search, pp. 269–274, ISTE (2007)
[Prestwich 09] Prestwich, S.: *Handbook of Satisfiability*, chapter 12: CNF Encodings, pp. 75–97, IOS Press (2009)
[Selman 92] Selman, B., Levesque, H. J., and Mitchell, D. G.: A New Method for Solving Hard Satisfiability Problems, in *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92)*, pp. 440–446 (1992)
[Soh 09] Soh, T., Inoue, K., Tamura, N., Banbara, M., and Nabeshima, H.: A SAT-based Method for Solving the Two-dimensional Strip Packing Problem, *Journal of Algorithms in Cognition, Informatics and Logic* (2009), to appear
[多賀 07] 多賀 明子, 田村 直之, 北川 哲, 番原 睦則：グリッド計算環境上でのショップ・スケジューリング問題の SAT 変換による解法, スケジューリング・シンポジウム 2007 講演論文集, pp. 109–114 (2007)
[Tamura 06] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M.: Compiling Finite Linear CSP into SAT, in *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP 2006), LNCS 4204*, pp. 590–603 (2006)
[田村 07] 田村 直之, 多賀 明子, 番原 睦則, 宋 剛秀, 鍋島 英知, 井上 克巳：ショップ・スケジューリング問題の SAT 変換による解法, スケジューリング・シンポジウム 2007 講演論文集, pp. 97–102 (2007)
[Tamura 08a] Tamura, N. and Banbara, M.: Sugar: a CSP to SAT Translator Based on Order Encoding, in *Proceedings of the Second International CSP Solver Competition*, pp. 65–69 (2008)
[Tamura 08b] Tamura, N., Tanjo, T., and Banbara, M.: System Description of a SAT-based CSP Solver Sugar, in *Proceedings of the Third International CSP Solver Competition*, pp. 71–75 (2008)
[Tamura 09] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M.: Compiling Finite Linear CSP into SAT, *Constraints*, Vol. 14, No. 2, pp. 254–272 (2009)
[Tanjo 08] Tanjo, T., Tamura, N., and Banbara, M.: Sugar++: a SAT-based Max-CSP/COP Solver, in *Proceedings of the Third International CSP Solver Competition*, pp. 77–82 (2008)
[The choco team 08] The choco team, : choco: an Open Source Java Constraint Programming Library, in *Proceedings of the Third International CSP Solver Competition*, pp. 7–13 (2008)
[Walsh 00] Walsh, T.: SAT v CSP, in *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP 2000)*, pp. 441–456 (2000)

〔担当委員：××○○〕

19YY 年 MM 月 DD 日　受理

著　者　紹　介

田村　直之（正会員）

1980 年神戸大学理学部物理学科卒業．1985 年同大学院自然科学研究科博士課程システム科学専攻修了．学術博士．1985 年日本 IBM 東京基礎研究所入社．1988 年神戸大学工学部勤務．2003 年より神戸大学学術情報基盤センター学術情報処理研究部門教授．制約プログラミング，SAT，線形論理，論理プログラミングなどに興味をもつ．日本ソフトウェア科学会，情報処理学会会員．

丹生　智也

2007 年神戸大学工学部情報知能工学科卒業．2009 年同大学大学院工学研究科博士課程前期課程情報知能学専攻修了．2009 年同研究科博士課程後期課程情報知能学専攻進学．制約プログラミング，論理プログラミング，SAT，擬似ブール制約などに興味をもつ．日本ソフトウェア科学会会員．

番原　睦則

1994 年神戸大学理学部数学科卒業．1996 年同大学大学院自然科学研究科博士課程前期課程数学専攻修了．1996 年国立奈良工業高等専門学校助手．1998 年同校講師．2003 年より神戸大学学術情報基盤センター学術情報処理研究部門講師．2007 年同校准教授．博士 (工学)．論理プログラミング，線形論理，制約プログラミング，SAT などに興味をもつ．日本ソフトウェア科学会，情報処理学会会員．