順序符号化と対数符号化を融合した制約充足問題のハイブリッド符号化

宋 剛秀　番原 睦則　田村 直之

本論文では制約充足問題 (CSP) から命題論理式の充足可能性判定問題 (SAT) への新しい符号化として，順序符号化と対数符号化を融合したハイブリッド符号化を提案する．順序符号化は効率的な制約伝播が特長であり小中規模のドメイン/アリティを持つ制約に対して有効な符号化であるが，それらが大きくなるにつれて符号化によって生成される SAT 節が膨大になる．一方，対数符号化は大規模なドメイン/アリティを持つ制約でも比較的少ない SAT 節で符号化できるが，桁上りのための推論が余分に必要なため制約伝播の効率はよくない．提案手法は両符号化の特長を活かし，小中規模のドメイン/アリティをもつ制約は順序符号化し，その他の制約については対数符号化する．人工的な問題 12 問と CSP ソルバー競技会のベンチマーク問題 1002 問を用いた実験の結果，提案手法は両符号化よりも高速に多くの問題を求解できることを確認した．さらにどちらの符号化でも解けない問題を解くことにも成功し，融合の相乗効果を確認した．

This paper proposes a new hybrid encoding of finite linear CSP to SAT integrating order and log encodings. The order encoding maintains bound consistency by unit propagation and works well for instances with small/middle domain sized variables and/or arity of constraints. The log encoding generates smaller CNF and is suitable for instances with larger domain sized variables, but its performance is not good in general because more inference steps are required to ripple carries. The proposed hybrid encoding takes advantages of both encodings by applying order encoding for constraints of small/middle sized domain/arity and log encoding for other constraints. We carried out experiments on 12 handmade instances and 1002 instances from CSP solver competition. In the result, our proposed hybrid encoding solved more number of instances within shorter time than each of order and log encodings. Moreover, we succeeded in solving instances which cannot be solved by both two encodings and confirm the synergy effect of the hybridization.

1　はじめに

制約充足問題 (CSP; Constraint Satisfaction Problem) は，与えられた制約を満たす解を探索する問題であり，人工知能などの分野における多くの組合せ問題は，CSP として定式化できることが知られている [22].

一方，与えられた命題論理の充足可能性判定問題 (SAT) や，その拡張である擬似ブール制約充足問題 (PB) について，それらを解くプログラムである SAT ソルバーや PB ソルバーの性能が大きく向上している [7] [16] [21]．それを背景として，CSP を SAT/PB に変換 (符号化) し，高速な SAT/PB ソルバーを用いて解く方法が成功を収めている [27] [29] [24].

CSP の命題論理への符号化としては，順序符号化 [8] [28]，対数符号化 [17] [12] など，様々な方法が提案されている．しかし，いずれの符号化も長所と短所が存在する．特に順序符号化は，2008 年および 2009 年の国際 CSP ソルバー競技会 [18] の複数部門で優勝した SAT 型制約ソルバー Sugar †1 に採用されており，多くの問題に対して優れた性能を示している [23] [4] [15]. しかし，順序符号化は変数のドメインサイズが大きい場合，生成される SAT 問題のサイズが巨大になり，解くことができないという問題点がある．一方，対数

A Hybrid Encoding of CSP to SAT Integrating Order and Log Encodings

Takehide Soh, Mutsunori Banbara, Naoyuki Tamura, 神戸大学情報基盤センター, Information Science and Technology Center, Kobe University.

†1 http://bach.istc.kobe-u.ac.jp/sugar/

符号化は，順序符号化に性能的には劣るが，大きな問題にも対応できるという利点を持つ．

したがって，複数の符号化を融合させた **ハイブリッド符号化** の研究が重要になる．特に，直接符号化 [9] [31] と順序符号化のハイブリッドについては論文 [13], BEE [19], meSAT [25] で研究されている．これらはいずれも各変数を直接符号化と順序符号化の両方で符号化し，さらにそれらの符号化を関連付けるためチャネリング制約と呼ばれるものを追加している．この方法 (ここではチャネリング方式と呼ぶ) は，直接符号化と順序符号化のハイブリッドについては成功したが，対数符号化とのハイブリッドには利用できない．なぜなら，対数符号化した変数を，直接符号化あるいは順序符号化と関連付けた場合，チャネリング制約が膨大になり，対数符号化の利点が失われてしまうからである．

そこで本論文では，チャネリング制約が不要な，順序符号化と対数符号化を融合した CSP のハイブリッド符号化を提案する．提案方法では CSP の各整数変数は順序符号化もしくは対数符号化のどちらか一つで符号化され，各制約は両方の符号化変数を含むことができる．どの変数にどの符号化を用いるかは任意に決めることができるため，どの問題・制約にどの符号化を適用するかを自由に切り替え可能なことも特長である．本論文では各変数の符号化方法を自動的に選択する方法についても考察を行い，ドメイン積を用いた方法の提案を行う．

提案方法は SAT 型 CSP ソルバー Diet-Sugar として実装を行った [†2]．Diet-Sugar は Sugar を基に実装され，XCSP [18] 形式で記述されたグローバル制約を含む CSP に対応している [18]．計算機実験では，Diet-Sugar を用いて順序符号化，対数符号化，提案するハイブリッド符号化の比較を行った．比較には 2 種類のベンチマークを用いた．一つ目は人工的な問題で順序符号化と対数符号化のハイブリッド符号化における相乗効果を確認するために用いた．二つ目はより広範な問題で評価を行うために CSP ソルバー競技会の 1002 問の問題を用いた．結果として，提案するハイブリッド符号化は順序符号化，対数符号化と比較して良い性能を示すことを確認した．

以下，2 節で CSP および 0-1 CSP, PB, SAT を定義する．3 節では，既存の符号化を統一的に説明する枠組みとして標準的ブール符号化を提案し，その枠組みを用いて対数符号化と順序符号化を定義する．4 節で，標準的ブール符号化の枠組みを用い，順序符号化と対数符号化を融合させたハイブリッド符号化を提案する．チャネリング制約が不要なハイブリッドを実現できている点が特長である．5 節では，提案したハイブリッド符号化を実装した Diet-Sugar を説明する．6 節では，国際 CSP ソルバー競技会のベンチマーク問題を用いて，ハイブリッド符号化を既存の順序符号化および対数符号化と比較・評価する．最後に，7 節で結論を述べる．

2 制約充足問題

本節では制約充足問題 (CSP; Constraint Satisfaction Problem) を定義し，さらにその一種として 0-1 制約充足問題 (0-1 CSP)，擬似ブール制約充足問題 (PB)，命題論理の充足可能性判定問題 (SAT) を定義する．

一般の制約充足問題では，整数，実数，集合等のドメイン上の様々な制約を対象としているが，本稿では整数有限領域上の線形の制約を対象とする．なお，実用的な制約充足問題の多くは，整数有限領域上の線形制約で表現できることが知られている．

$\mathcal{X}$ を **変数** の集合とする (以下 x, y, x_1, y_1 等で変数を表す)．$x_1, x_2, \ldots, x_n$ が変数，$a_1, a_2, \ldots, a_n$ および c が整数定数の時，$\sum_{i=1}^{n} a_i x_i$ の形の式を **線形和** (linear summation) と呼び，$\sum_{i=1}^{n} a_i x_i \# c$ の形の式を **線形比較** (linear comparison) と呼ぶ (以下 S, S_1 等で線形和を表し， E, E_1 等で線形比較を表す)．ただし $a_i \neq 0$ かつ x_i はすべて異なると仮定する．線形比較 $S \geq c$ の否定 $\neg(S \geq c)$ を $-S \geq -c+1$ で定義する．なお，線形比較の比較演算子が $=, \neq, \leq$ 等の場合も簡単に $\geq$ の線形比較に書き換えることができるため，比較演算子を $\geq$ に限定しても一般性を失わない．線形和もしくは線形比較中に含まれる変数の数を **アリティ** (ality) と呼ぶ．線形比較の有限集合

†2 http://kix.istc.kobe-u.ac.jp/~soh/dsugar/

を **制約** (constraint) と呼び，制約の有限集合を **制約集合** (constraint set) と呼ぶ．以下 C, C_1 等で制約を表し，$\boldsymbol{C}$ 等の斜体太字で制約集合を表す．

定義 1 **制約充足問題** ***(CSP**; Constraint Satisfaction Problem)* は，以下を満たす組 $(X, Dom, \boldsymbol{C})$ である．

- X は **変数** の有限集合 $(X \subset \mathcal{X})$.
- Dom は各変数の取りうる値の集合である **ドメイン** を定める関数．ここでは，ドメインは整数の有界の範囲とする．
- $\boldsymbol{C}$ は X 上の線形比較から成る制約集合.

制約集合 $\boldsymbol{C} = \{C_1, \ldots, C_n\}$ は制約 C_i の連言を意味し，$C_1 \wedge \cdots \wedge C_n$ あるいは C_i の列で表すことがある．また，制約 $C = \{E_1, \ldots, E_m\}$ は線形比較 E_j の選言を意味し，$E_1 \vee \cdots \vee E_m$ で表すことがある．すなわち制約集合は，線形比較の CNF 式 (Conjunctive Normal Form 式) である．

線形比較 E 中に現れる変数の集合を $Var(E)$ で表す．制約 C についても同様に $Var(C)$ で表示する．変数 x のドメイン $Dom(x)$ の下限と上限を，それぞれ $lb(x)$ と $ub(x)$ で表す．同様に線形和 S の下限と上限を，それぞれ $lb(S)$ と $ub(S)$ で表す．

CSP $(X, Dom, \boldsymbol{C})$ において，$Dom(x) \subset \{0, 1\}$ の場合，x を **ブール変数** と呼ぶ (以下 p, q, p_1, q_1 等でブール変数を表す).

1 つのブール変数から成る線形比較を **リテラル** (literal) と呼ぶ (以下 L, L_1 等でリテラルを表す)，リテラルでない線形比較を **複雑な線形比較** と呼ぶ．

定義 2 複雑な線形比較 *(*リテラルでない線形比較*)* を高々 *1* つ含む制約を **単純な制約** *(simple constraint)* と呼び，すべての制約が単純な *CSP* を **単純 *CSP*** と呼ぶ．

$\{p, \neg q\}$, $\{p_1, p_2, x - y \geq 0\}$ は単純な制約の例である．

定義 3 すべての変数がブール変数の制約を ***0-1* 制約** と呼び，すべての制約が *0-1* 制約の *CSP* を ***0-1* *CSP*** と呼ぶ．

$\{p_1 - p_2 \geq 0, p_2 - p_3 \geq 0\}$ は 0-1 制約の例である．

定義 4 *1* つの線形比較からなる *0-1* 制約を **擬似ブール制約** *(*あるいは ***PB* 制約***)* と呼び，すべての制約が擬似ブールの *CSP* を **擬似ブール制約充足問題** ***(PB)*** と呼ぶ．

$\{p_1 + 2p_2 \geq 1\}$ は擬似ブール制約の例である．擬似ブール制約は単純 0-1 制約，PB は単純 0-1 CSP である．

定義 5 すべての線形比較がリテラルの制約を **節** *(clause)* と呼び，すべての制約が節の *CSP* を **命題論理の充足可能性判定問題** ***(SAT)*** と呼ぶ．

$\{p, \neg q\}$ は節の例である．節は単純 0-1 制約，SAT は単純 0-1 CSP である．

3 標準的ブール符号化としての対数符号化と順序符号化

本節の前半では，**標準的ブール符号化** の定義を行う．この枠組では SAT 符号化は与えられた変数集合に対して部分的に適用され，これを利用することでチャネリング制約を用いることなく SAT 符号化のハイブリッドを行うことが可能になる．本節の後半では，対数符号化と順序符号化を標準的ブール符号化を用いて定式化する．

3.1 標準的ブール符号化

CSP から CSP への **変換** (あるいは **符号化**) τ において，元の CSP P と変換後の CSP P' の充足可能性が一致する時，変換 τ は **充足同値** (equi-satisfiable) であるという．変換 τ_1, τ_2 が充足同値な時，合成 $\tau_1 \circ \tau_2$ [†3] もまた充足同値な変換である．

与えられた変数集合に関して部分的に符号化を行うように対数符号化と順序符号化を定義するために，以下の **標準的ブール符号化** を導入する．

定義 6 与えられた変数の部分集合 V に関する **標準的ブール符号化** は以下を満たす組 $(U, \boldsymbol{A}, T)$ である．ただし，CSP $P = (X, Dom, \boldsymbol{C})$ ， $V \subseteq X$ とする．

- U は符号化によって追加される変数の集合を表す． $X \cap U = \emptyset$ とする．
- $\boldsymbol{A}$ は追加する制約集合を表す.
- T は与えられた線形比較を制約集合に変換する関数である．

†3 $((f \circ g)(x) = g(f(x)))$

この標準的なブール符号化によって得られる CSP $P' = (X', Dom', \boldsymbol{C}')$ は以下になる．

$$\begin{aligned} X' &= (X \setminus V) \cup U \\ Dom'(x) &= \begin{cases} Dom(x) & (x \in X \setminus V) \\ \{0,1\} & (x \in U) \end{cases} \\ \boldsymbol{C}' &= \boldsymbol{A} \cup \bigwedge_{C \in \boldsymbol{C}} \bigvee_{E \in C} T(E) \end{aligned}$$

ここで注意したいのは，$\bigvee_{E \in C} T(E)$ の組合せが場合によっては膨大になることである．例えば複雑な線形比較のみで構成される制約 $C = \{E_1, ..., E_n\}$ が与えられた時に，C を標準的ブール符号化することを考える．制約 C は素朴に考えると $\prod_{i=1}^{n} |T(E_i)|$ 個の制約になる．しかし，Tseitin 変換[30] を用いると制約の増加を抑えながら CSP から単純 CSP への変換が可能になる．

例 1 複雑な線形比較で構成される制約 $C = \{E_1, E_2\}$ が与えられたとする．*Tseitin* 変換により，充足同値な単純制約の集合 $\{\{p, q\}, \{\neg p, E_1\}, \{\neg q, E_2\}\}$ が得られる．元の制約と比較すると符号化後の制約数は $|T(E_1)| \times |T(E_2)|$ から $|T(E_1)| + |T(E_2)| + 1$ へと削減できることが分かる．

3.2 対数符号化

対数符号化 (もしくは 2 進符号化) は CSP から 0-1 CSP への充足同値な変換である．元の CSP の変数 x に対し，$x - lb(x)$ の 2 進表現の i ビット目を表す新しいブール変数 $p_{x,i}$ を導入し，x をそれらの重み付き和で表す[17][12].

変数集合 V に対する対数符号化 τ_V^{L} を以下の $(U_V^{\mathrm{L}}, \boldsymbol{A}_V^{\mathrm{L}}, T_V^{\mathrm{L}})$ で表す．ただし，CSP $P = (X, Dom, \boldsymbol{C})$, $V \subseteq X$ とする．

$$\begin{aligned} U_V^{\mathrm{L}} &= \{p_{x,i} \mid x \in V,\ 0 \le i \le m_x\} \\ \boldsymbol{A}_V^{\mathrm{L}} &= \{\{x\sigma \le ub(x)\} \mid x \in V\} \\ T_V^{\mathrm{L}}(E) &= \{\{E\sigma\}\} \\ \sigma &= \{x \mapsto lb(x) + \textstyle\sum_{i=0}^{m_x} 2^i p_{x,i} \mid x \in V\} \end{aligned}$$

ここで記号 $\mapsto$ は変数集合から線形和への写像を表す．$E\sigma$ は線形比較 E において変数 x を線形和 $\sigma(x)$ で置換した結果を表す．

例 2 変数集合 $X = \{x, y\}$, ドメイン $Dom(x) = \{1, 2, 3\}$, 制約 $\boldsymbol{C} = \{\{E\}\} = \{\{x - y \ge 1\}\}$ から構成される *CSP* $(X, Dom, \boldsymbol{C})$ と変数集合 $V = \{x\}$ を考える．τ_V^L によって以下が得られる．

$$\begin{aligned} U_V^L &= \{p_{x,0}, p_{x,1}\} \\ \boldsymbol{A}_V^L &= \{\{1 + p_{x,0} + 2p_{x,1} \le 3\}\} \\ T_V^L(\{E\}) &= \{\{1 + p_{x,0} + 2p_{x,1} - y \ge 1\}\} \end{aligned}$$

3.3 順序符号化

順序符号化 (order encoding) は，CSP から 0-1 CSP への充足同値な変換である．元の CSP の変数 x および値 $d \in Dom(x) \setminus \{lb(x)\}$ に対し $x \ge d$ を意味する新しいブール変数 $p_{x \ge d}$ を導入する[8][28] (引用論文では $x \ge d$ ではなく $x \le d$ にブール変数を対応させている).

例えば $Dom(x) = Dom(y) = \{1..5\}$ なる変数 x, y および制約 $x - y \ge 0$ の符号化を考える．

- 変数 x について以下の 4 つのブール変数を導入する ($p_{x \ge 1}$ は $x \ge 1$ が常に真のため不要)．変数 y についても同様とする．

  $$p_{x \ge 2} \quad p_{x \ge 3} \quad p_{x \ge 4} \quad p_{x \ge 5}$$

- 導入したブール変数に対し，以下の制約を導入する．変数 y についても同様とする．

  $\{p_{x \ge 2}, \neg p_{x \ge 3}\}, \{p_{x \ge 3}, \neg p_{x \ge 4}\}, \{p_{x \ge 4}, \neg p_{x \ge 5}\}$

  $x = 1$ は $p_{x \ge d}$ のすべてが偽の時に対応し，$x = 5$ はすべてが真の時に対応する．

- この時 $x - y \ge 0$ は，以下のように符号化できる．

  $$\{\{p_{x \ge d}, \neg p_{y \ge d}\} \mid 1 < d \le 5\}$$

変数 x に対する順序符号化 τ_x^{o} を標準的変換として以下の $(U, V, D, \boldsymbol{A}, \|\cdot\|)$ で定める (τ_x^{o} の o は order を表す)．変数集合 V に対する順序符号化 τ_V^{O} を以下の $(U_V^{\mathrm{O}}, \boldsymbol{A}_V^{\mathrm{O}}, T_V^{\mathrm{O}})$ で表す．ただし，CSP $P = (X, Dom, \boldsymbol{C})$, $V \subseteq X$ とする．

$$\begin{aligned} U_V^{\mathrm{O}} &= \{p_{x \ge d} \mid x \in V,\ lb(x) < d \le ub(x)\} \\ \boldsymbol{A}_V^{\mathrm{O}} &= \{\{p_{x \ge d}, \neg p_{x \ge d+1}\} \\ &\qquad \mid x \in V,\ lb(x) < d < ub(x)\} \\ T_V^{\mathrm{O}}(E) &= \begin{cases} \{\{E\}\} & (Var(S) \cap V = \emptyset) \\ T'(E) & (Var(S) \cap V \ne \emptyset) \end{cases} \end{aligned}$$

ここで $E = \sum_{i=1}^{n} a_i x_i$ と $T'(E)$ は以下のように定義される．

$$T'(E) = \begin{cases} \{\{p_{x_1 \geq \lceil c/a_1 \rceil}\}\} & (n=1,\ x_1 \in V,\ a_1 > 0) \\ \{\{\neg p_{x_1 \geq \lfloor c/a_1 \rfloor + 1}\}\} & (n=1,\ x_1 \in V,\ a_1 < 0) \\ \bigwedge_{d \in Dom(x_i)} (\{\{p_{x_i \geq d+1}\}\} \vee T_V^{\mathrm{O}}(\sum_{j \neq i} a_j x_j \geq c - a_i d)) & (n > 1,\ x_i \in V,\ a_i > 0) \\ \bigwedge_{d \in Dom(x_i)} (\{\{\neg p_{x_i \geq d}\}\} \vee T_V^{\mathrm{O}}(\sum_{j \neq i} a_j x_j \geq c - a_i d)) & (n > 1,\ x_i \in V,\ a_i < 0) \end{cases}$$

上述の定義において，$d \leq lb(x)$ の時 $p_{x \geq d} = 1$, $d > ub(x)$ の時 $p_{x \geq d} = 0$ とする．

例 3 変数集合 $X = \{x, y\}$, ドメイン $Dom(x) = \{0, 1, 2\}$, 制約 $\boldsymbol{C} = \{\{E\}\} = \{\{x + y \geq 0\}\}$ から構成される $CSP\ (X, Dom, \boldsymbol{C})$ と変数集合 $V = \{x\}$ を考える．τ_V^O によって以下が得られる．

$$\begin{aligned} U_V^O &= \{p_{x \geq 1}, p_{x \geq 2}\} \\ \boldsymbol{A}_V^O &= \{\{p_{x \geq 1}, \neg p_{x \geq 2}\}\} \\ T_V^O(\{E\}) &= \{\{p_{x \geq 1}, y \geq 0\}, \{p_{x \geq 2}, y \geq -1\}, \\ &\quad \{y \geq -2\}\} \end{aligned}$$

3.4 0-1CSP から SAT への符号化

ここまで説明してきた対数符号化，順序符号化を用いることで, CSP をブール変数しか含まない 0-1CSP へと符号化できる．さらに 3.1 節で説明した Tseitin 変換を適用することで単純 0-1CSP が得られる．単純 0-1CSP は Big-M 法 [14] により PB へと変換することができる．例えば，ブール変数 p と線形和 S から構成される制約 $\{p, S \geq c\}$ を考える．この制約は Big-M 法により $(c - lb(S))p + S \geq c$ へと変換される．ここで $lb(S)$ は線形和の下限値を表す．この制約は $p = 1$ の時に明らかに充足可能であり，$p = 0$ の時は $S \geq c$ と同値になる．

以上により，CSP は単純 CSP，単純 0-1 CSP を経て PB に変換できる．PB に変換されれば，これまで数多く提案されている PB を SAT への符号化方法を利用することができる [10] [3] [1] [26] ．他の方法としては直接 PB ソルバーを使う方法であり [5] [11] [10] ，この場合は PB から SAT への符号化は必要ない．

4 順序符号化と対数符号化を融合したハイブリッド符号化

本節では前節までで定義した変数の部分集合に対する対数符号化と順序符号化を用いてそれらを融合したハイブリッド符号化を説明する．

4.1 ハイブリッド符号化

CSP $P = (X, Dom, \boldsymbol{C})$ と X の分割 $\{V_O, V_L\}$ が与えられた時，順序符号化と対数符号化を融合したハイブリッド符号化ハイブリッド符号化 $\tau_X^{\mathrm{H}} = \tau_{V_L}^{\mathrm{L}} \circ \tau_{V_O}^{\mathrm{O}}$ は以下のように定義される．

$$\begin{aligned} U_X^{\mathrm{H}} &= U_{V_L}^{\mathrm{L}} \cup U_{V_O}^{\mathrm{O}} \\ A_X^{\mathrm{H}} &= A_{V_L}^{\mathrm{L}} \cup A_{V_O}^{\mathrm{O}} \\ T_X^{\mathrm{H}}(E) &= T_{V_O}^{\mathrm{O}}(T_{V_L}^{\mathrm{L}}(E)) \end{aligned}$$

ここで対数符号化と順序符号化の合成は可換であることに注意されたい．すなわち $\tau_{V_L}^{\mathrm{L}} \circ \tau_{V_O}^{\mathrm{O}} = \tau_{V_O}^{\mathrm{O}} \circ \tau_{V_L}^{\mathrm{L}}$ が成り立つ．

例 4 変数集合 $X = \{x, y\}$, ドメイン $Dom(\cdot) = \{0, 1, 2\}$, 制約 $\boldsymbol{C} = \{\{E\}\} = \{\{x + y \geq 0\}\}$ から構成される $CSP\ (X, Dom, \boldsymbol{C})$ を考える．また対数符号化，順序符号化する変数集合をそれぞれ $V_L = \{x\}$, $V_O = \{y\}$ とする．ハイブリッド符号化 τ_X^H によって以下が得られる．

$$\begin{aligned} U_X^H &= \{p_{x,0}, p_{x,1}\} \cup \{p_{y \geq 1}, p_{y \geq 2}\} \\ A_X^H &= \{\{2p_{x,1} + p_{x,0} \leq 2\}\} \cup \{\{p_{y \geq 1}, \neg p_{y \geq 2}\}\} \\ T_X^H(E) &= T_{V_O}^O(T_{V_L}^L(E)) \\ &= T_{V_O}^O(\{2p_{x,1} + p_{x,0} + y \geq 0\}) \\ &= \{\{2p_{x,1} + p_{x,0} \geq 0, p_{y \geq 1}\}, \\ &\quad \{2p_{x,1} + p_{x,0} \geq -1, p_{y \geq 2}\}, \\ &\quad \{2p_{x,1} + p_{x,0} \geq -2\}\} \end{aligned}$$

4.2 変数の分類

提案するハイブリッド符号化では，順序符号化すべき変数，対数符号化すべき変数をどのように分類して X の分割 V_L と V_O を作るのかが重要になる．分類の方法は大きく二つある：一つ目は変数に着目する方法でドメインサイズが分類の指標になる．二つ目は変数のドメインサイズと制約のアリティの両方を用いる方法である．一つ目の変数だけに着目する方法は直感

的だが順序符号化によって生成される節数が膨大になる場合がある．

例えば 3 つの整数変数 $x, y \in \{1, \ldots, 1000\}$, $z \in \{1, \ldots, 10000\}$ と二つの制約 $x \leq z$ と $x + y \leq z$ が与えられたとする．ドメインサイズのみを指標とし，閾値を 1000 に設定した場合，x と y は順序符号化に分類され z は対数符号化に分類される．この場合，一つ目の制約 $x \leq z$ の符号化節数は 12993 となるが，二つ目の制約 $x + y \leq z$ の符号化節数は 1011993 であり，これは明らかに SAT ソルバーが扱える節数の限界に近い．この例からも明らかなように，順序符号化によって得られる節数は制約のアリティが増えるに伴って急激に増加する．このような増加を避けるためにドメインサイズだけでなく制約のアリティも考慮した指標が必要になる．

そこで本論文ではドメインサイズの積，**ドメイン積** を変数の分類の指標に使うことを提案する．線形比較 $E = \sum_{i=1}^{n} a_i x_i \geq c$ に対してドメイン積 $DP(E)$ は次のように定義される．

$$DP(E) = \prod_{i=1}^{n-1} |Dom(x_i)|$$

ここで変数はドメインサイズの昇順にソートされているものとする．順序符号化による符号化節数の上限は d をドメインサイズ，n をアリティとした時に $O(d^{n-1})$ に与えられることから，ドメイン積は順序符号化によって生成される節数の概算にもなっている．この指標に加えて，PB 制約は 0-1 変数しか含まないことから対数符号化を用いるのが妥当なので，PB 制約に含まれる変数は対数符号化するのが適当である．

提案する分類方法を以下に定義する．CSP $(X, Dom, \boldsymbol{C})$ と閾値 θ が与えられた時，ハイブリッド符号化において対数符号化する変数の部分集合は以下のように定義される．

$V_L = \{v \in Var(E) \mid$
$DP(E) > \theta$ or E は 0-1 線形比較, $E \in c$, $c \in \boldsymbol{C}\}$.

ハイブリッド符号化は以下のように動作する．

- $V_L = \emptyset$ の時は順序符号化と同じ
- $V_L = X$ の時は対数符号化と同じ
- $\emptyset \subsetneq V_L \subsetneq X$ の時，順序符号化と対数符号化のハイブリッド

5 Diet-Sugar の実装

提案するハイブリッド符号化を用いた SAT 型 CSP ソルバー Diet-Sugar を Sugar を基にして実装した．外延的制約を除いて，Diet-Sugar は *alldifferent* などのグローバル制約を含む全ての XCSP 形式[18] に対応可能である．また Sugar の専用 CSP 言語にも対応している．Diet-Sugar は以下の手順で CSP の解を計算する:

1. CSP を定義 1 の形式に変換し, 3.1 節に記載した Tseitin 変換によって単純 CSP を得る．
2. 4 節に記載したハイブリッド符号化によって単純 CSP を単純 0-1CSP に変換する．
3. 単純 0-1CSP を 3.4 節に記載した方法で SAT に変換する．

今回 Diet-Sugar のバックエンドソルバーには最新バージョンの minisat+ (ver. 1.1) とそのデフォルトの SAT ソルバーである minisat (ver. 2.2) を用いた．両ソフトウェアは github[†4] から入手可能である．この他に Diet-Sugar では minisat+ を PB から SAT への符号化器として用いることで任意の SAT ソルバーを利用可能である [2] [6] [11] [20] [5].

6 計算機実験

本節では提案するハイブリッド符号化の求解性能を順序符号化と対数符号化と比較する．実験条件を揃えるために三つの符号化は 5 節の手順 2 で切り替えた．計算機は CPU として Xeon E5 3.7GHz，メモリを 32GB 搭載したものを使用した．

比較には二つのベンチマークセットを用いた．一つ目のベンチマークは人工的な問題であり，ハイブリッドの相乗効果を確認するために用いた．二つ目のベンチマークは 2009 年の CSP ソルバー競技会の 1002 問であり，多種の問題を用いた広範な比較を行うために用いた．

本実験の前にドメイン積の閾値 θ を決定するための予備実験を行った．θ として 1024, 2048, 4096, 8192 を実験した結果，最も性能が良かった 4096 を以降の

†4 https://github.com/niklasso

実験では使用する．

6.1 人工的な問題を用いた評価

本節では, 順序符号化に適した制約，対数符号化に適した制約の両方の制約を含むような人工的な問題を考え，提案するハイブリッド符号化による相乗効果を確認する．

6.1.1 問題定義

整数のパラメータ n, d と，ドメイン $\{0, \ldots, d\}$ を持つような整数変数 x_i, y_i $(1 \leq i \leq n)$ が与えられた時に，以下のような制約をもつ CSP を考える．

$$\bigwedge_{i=1}^{n} x_i \geq y_i \ \wedge \ \sum_{i=1}^{n} x_i < \sum_{i=1}^{n} y_i$$

x の和が y の和以上にならない制約からこの問題は任意の n と d に対して充足不能である．このベンチマーク問題を生成する際には線形和 $x_1 + \cdots + x_n$ のアリティが 3 になるまで新しい変数を導入して再帰的に分割した (y についても同様)．例えば $n = 4$ の時に問題は以下のようになる．

$$\begin{array}{llll} x_1 \geq y_1 & x_1 + x_2 = u_1 & u_1 + u_2 = s & s < t \\ x_2 \geq y_2 & x_3 + x_4 = u_2 & & \\ x_3 \geq y_3 & y_1 + y_2 = v_1 & v_1 + v_2 = t & \\ x_4 \geq y_4 & y_3 + y_4 = v_2 & & \end{array}$$

ここで u と v は部分和を表す補助変数であり，s と t は全体の和を表している．

この CSP は様々なドメイン積を一つの問題中に含む．例えば，上述の例で $d = 10$ とするとドメイン積 $DP(x_1 + x_2 = u_1)$ は 100 であるのに対して $DP(u_1 + u_2 = s)$ は 10000 になる．また n と d を変更することで問題の性質を制御することができる．例えば d が大きくなると符号化節数が大きくなり対数符号化に適した問題となる．一方，n が大きくなると x_i への値割当が $\lfloor \log_2 n \rfloor$ 個の補助変数を介して s へと伝播する必要があるため，伝播に優れた順序符号化に適した問題となる．

今回は以下二つのパラメータ集合を用いてベンチマークを生成し実験を行った．制限時間は 300 秒である．

- 一つ目のパラメータ集合は $n = 4$ で固定とし，d を $50, 100, 150, 200, 250, 300$ で変化させた．ドメインが大きいため対数符号化が効果的であると考えられる．
- 二つ目のパラメータ集合は $n = 8$ で固定とし，d を $10, 20, 30, 40, 50, 60$ で変化させた．ドメインがあまり大きくなく，n が比較的大きいため問題を解くために伝播が必要で順序符号化が効果的であると考えられる．

6.1.2 実験結果

表 1 は CPU 時間，各符号化で生成された節の数, SAT ソルバーによる求解過程で生じた矛盾の回数を比較したものである．以降，実験結果の表や図では順序符号化を Order，対数符号化を Log，提案するハイブリッド符号化を Hybrid と参照する．

表より対数符号化は非常にコンパクトな符号化を行っており，$n = 4$ の場合には $d = 300$ になっても 10 秒以内に問題を解いていることが分かる．しかしながら，$n = 8$ では対数符号化の求解性能が落ちていることも分かる．理由は対数符号化では桁上げのための推論手続きが必要であり，より多くの値の伝播が必要な $n = 8$ で解を求めることが難しくなったのだと考えられる．

対照的に，順序符号化は $n = 8$ の時に対数符号化よりも少ない矛盾回数でより多くの問題を解いていることが分かる．しかしながら，順序符号化では $n = 4$ の時にドメイン d の大きさが 250 を超えると問題が解けなくなった．原因として節数が膨大になったことが挙げられる．実際，$n = 4$ で $d = 300$ の時に順序符号化で生成される節数は 2179179 であり，これは対数符号化の節数 1677 の 1000 倍以上になっている．

提案するハイブリッド符号化は適切にそれぞれの符号化の長所を引き継いでおり，両符号化で解けた問題を全て解くことができた．また引き継ぐだけでなく，$n = 8$ の時にハイブリッド符号化は順序符号化，対数符号化のいずれよりも少ない矛盾回数で問題を解くことに成功しており，ハイブリッドの相乗効果を確認できた．特に $n = 8$ と $d = 60$ の問題を制限時間内に解くことができたのはハイブリッド符号化のみであった．

表 1　人工的な問題における CPU 時間，節数，矛盾回数の比較

n	d	CPU 時間 (秒)			符号化節数			矛盾回数		
		Order	Log	Hybrid	Order	Log	Hybrid	Order	Log	Hybrid
4	50	4.40	1.05	4.69	63,179	1,088	33,644	50,026	67,154	8,078
4	100	19.26	1.86	1.86	246,379	1,283	1,283	156,794	168,198	168,198
4	150	85.33	2.35	2.35	549,579	1,482	1,482	422,925	218,686	218,686
4	200	257.63	4.14	4.13	972,779	1,470	1,470	935,133	436,850	436,850
4	250	TO	2.47	2.49	1,515,979	1,438	1,438	—	249,755	249,755
4	300	TO	6.88	6.90	2,179,179	1,677	1,677	—	716,163	716,163
8	10	1.29	1.24	1.28	12,923	1,691	12,923	8,508	61,529	8,508
8	20	4.43	31.80	3.95	48,283	2,118	29,774	64,066	2,339,531	11,722
8	30	35.91	87.11	7.80	106,043	1,998	62,986	536,078	6,498,426	26,737
8	40	45.63	TO	12.05	186,203	2,529	45,177	500,552	—	87,173
8	50	131.39	TO	22.19	288,763	2,487	67,599	1,150,698	—	142,961
8	60	TO	TO	35.95	413,723	2,441	94,633	—	—	190,635

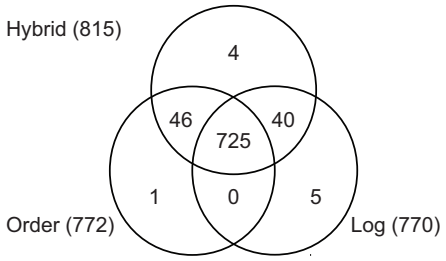

図 1　求解できた問題を表すベン図

6.2　CSP ソルバー競技会の問題を用いた評価

6.2.1　ベンチマークの説明

本節では 2009 年の CSP ソルバー競技会で用いられた問題 1002 問についての比較結果を説明する．1002 問の問題は競技会で使われた問題の全体集合から内包的制約であり連続ドメインを持つ全ての問題を抽出した．ただし，PB の問題　(363 問) については取り除いた．理由は 0-1 変数のみで構成される PB 制約には明らかに対数符号化が適しており，PB 制約には対数符号化が選ばれるようハイブリッド符号化が動作するためである．実験で使用した 1002 問のドメインサイズは 1〜20001，アリティは 1〜50 に分布しており，このことから様々な種類の問題が含まれていることが分かる．全ての実験は 900 秒の時間制限で実行した．

6.2.2　実験結果

図 1 に示すベン図において，中心の数字 725 は各符号化で共通に解けた問題数を表している．同様に左側の数 $46+1$ は順序符号化で解けて対数符号化で解けない問題，右側の数 $40+5$ は対数符号化で解けて順序符号化で解けない問題を表している．ハイブリッド符号化は，それらの 93% (86/92) を解いたうえで，さらに両方の符号化で解けなかった 4 問の問題を解くことに成功した．これらのことより，提案するハイブリッド符号化は CSP 競技会の問題 1002 問において，順序符号化と対数符号化の融合を高いレベルで実現できたといえる．

表 2 は，それぞれの符号化によって解けた問題の数をシリーズ毎に示している．太字で示されたシリーズはハイブリッド符号化の閾値をまたぐ複数のドメインサイズの制約を含む問題である．すなわちこれらのシリーズでハイブリッド符号化は順序符号化，対数符号化の両方も利用する．図 1 は，それぞれの符号化で解けた問題の包含関係を示したベン図である．図 2 は，それぞれの符号化で解けた問題数と CPU 時間の関係を記載したカクタスプロットである．

順序符号化と対数符号化が 772 問と 770 問を解いたのに対して，提案するハイブリッド符号化は最も多い 815 問を解いた．特に，“Fischer”，“Square

表 2　シリーズ毎の求解数

シリーズ名 (問題の数)		Ord.	Log	Hyb.
2D Strip Packing	(20)	**8**	7	**8**
All Interval Series	(15)	**9**	**9**	**9**
BMC	(15)	**15**	**15**	**15**
Chessboard Col.	(15)	**12**	**12**	**12**
C. Job-Shop	(10)	**5**	4	**5**
Domino	(10)	9	**10**	9
Fischer	(25)	17	23	**24**
Golomb Ruler	(28)	**23**	18	**23**
Graph Coloring	(141)	**98**	97	**98**
Haystacks	(15)	**3**	1	**3**
Job-Shop	(76)	**67**	61	**67**
Knights	(10)	7	**10**	9
Langford	(43)	**27**	24	**27**
Latin Square	(10)	**9**	5	**9**
Magic Square	(18)	8	**13**	**13**
Multi Knapsack	(6)	4	**6**	**6**
NengFa	(7)	**7**	6	**7**
Open-Shop	(75)	**71**	60	**71**
Square Packing	(74)	56	56	**57**
Pigeons	(29)	**23**	**23**	**23**
Primes	(76)	44	69	**70**
Quasigroup Exist.	(5)	**5**	**5**	**5**
Queens	(16)	11	**12**	11
Queens-Knights	(9)	**9**	**9**	**9**
RCPSP	(78)	**78**	**78**	**78**
Rader Surveillance	(65)	**65**	**65**	**65**
Ramsey	(16)	**10**	**10**	**10**
Schurr's Lemma	(10)	**9**	8	**9**
Super-solutions	(85)	**63**	54	**63**
Total	(1002)	772	770	**815**

Packing", "Primes" ではハイブリッド符号化がどちらの単一符号化でも解けない問題を解くことに成功した．

ここで注意したいのは，CSP の SAT 符号化に必要な時間は PB 制約が本質的に節形式になっている場合でも PB を SAT に変換する時間を含むということである．このオーベーヘッドを調べるために，同じベンチマークに対して，Sugar を用いて実験を行った．結果として，求解数は 787 でオーバーヘッドを抱えている提案方法による求解数が多かった．この差は既に節形式になっている PB から SAT への符号化のオーバーヘッドが小さくなれば，より大きくなると考えられる．

7　おわりに

本論文では，順序符号化と対数符号化を融合した CSP のハイブリッド符号化を提案した．まず CSP の変数集合毎に異なる符号化を適用可能にする標準的ブール符号化の枠組みを提案し，次に具体的なハイブリッド符号化の説明を行った．提案方法では与えられた CSP の変数集合を対数符号化するものと順序符号化するものに分類する必要があるが，分類方法としてドメイン積を用いた方法を提案した．

提案方法は公開ソフトウェア Diet-Sugar として実装し，二つのベンチマークを使って評価を行った．人工的なベンチマークを用いた評価では，提案するハイブリッド符号化が順序符号化，対数符号化と比較してより少ない CPU 時間と矛盾回数で問題を解くことができることを示した．特に $n = 8, d = 60$ の問題では，どちらの符号化でも解けない問題を解くことに成功した．二つ目のベンチマークでは，順序符号化で解けて対数符号化で解けない問題，対数符号化で解けて順序符号化で解けない問題の 93%を解くこと，またどちらの符号化でも解けなかった 4 問をハイブリッド符号で解くことに成功した．最終的に求解した問題数は順序符号化が 772 問，対数符号化が 770 問であるのに対して提案するハイブリッド符号化は 815 問と最も多くの問題を解いた．これらの結果から提案方法は高いレベルで二つの符号化のハイブリッド化を実現できているといえる．今後の研究課題として他の既存の SAT 符号化を実装し，ハイブリッド符号化に組み込むことは重要だと考えている．

参 考 文 献

[1] Abío, I., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E., and Mayer-Eichberger, V.: A New Look at BDDs for Pseudo-Boolean Constraints, *Journal of Artificial Intelligence Research*,

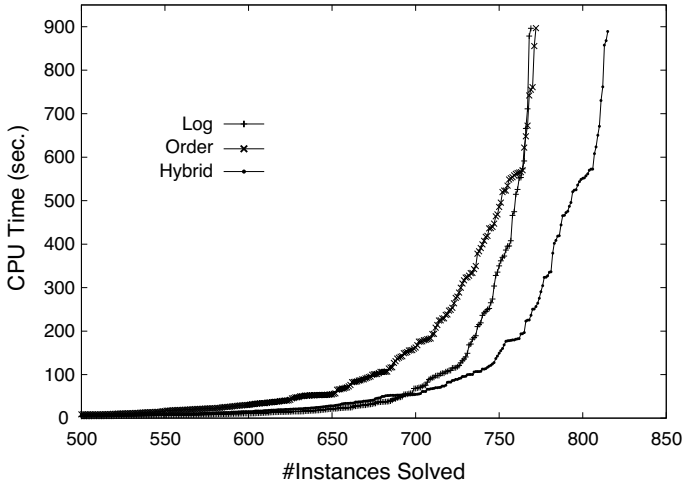

図 2　解けた問題の数と CPU 時間のカクタスプロット

Vol. 45(2012), pp. 443–480.

[2] Audemard, G. and Simon, L.: Predicting Learnt Clauses Quality in Modern SAT Solvers, *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 2009, pp. 399–404.

[3] Bailleux, O., Boufkhad, Y., and Roussel, O.: A Translation of Pseudo Boolean Constraints to SAT, *Journal on Satisfiability, Boolean Modeling and Computation*, Vol. 2, No. 1-4(2006), pp. 191–200.

[4] Banbara, M., Matsunaka, H., Tamura, N., and Inoue, K.: Generating Combinatorial Test Cases by Efficient SAT Encodings Suitable for CDCL SAT Solvers, *Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-17), LNCS 6397*, 2010, pp. 112–126.

[5] Berre, D. L. and Parrain, A.: The Sat4j Library, release 2.2, *Journal on Satisfiability, Boolean Modeling and Computation*, Vol. 7, No. 2-3(2010), pp. 59–64.

[6] Biere, A.: Lingeling Essentials, A Tutorial on Design and Implementation Aspects of the the SAT Solver Lingeling, *POS-14. Fifth Pragmatics of SAT workshop, a workshop of the SAT 2014 conference, part of FLoC 2014 during the Vienna Summer of Logic, July 13, 2014, Vienna, Austria*, 2014, pp. 88.

[7] Biere, A., Heule, M., van Maaren, H., and Walsh, T.(eds.): *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications (FAIA), Vol. 185, IOS Press, 2009.

[8] Crawford, J. M. and Baker, A. B.: Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems, *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 1994)*, 1994, pp. 1092–1097.

[9] de Kleer, J.: A Comparison of ATMS and CSP Techniques, *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI 1989)*, 1989, pp. 290–296.

[10] Eén, N. and Sörensson, N.: Translating Pseudo-Boolean Constraints into SAT, *Journal on Satisfiability, Boolean Modeling and Computation*, Vol. 2, No. 1-4(2006), pp. 1–26.

[11] Gebser, M., Kaufmann, B., Neumann, A., and Schaub, T.: Conflict-Driven Answer Set Solving, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007, pp. 386–392.

[12] Gelder, A. V.: Another Look at Graph Coloring via Propositional Satisfiability, *Discrete Applied Mathematics*, Vol. 156, No. 2(2008), pp. 230–243.

[13] Gent, I. P. and Nightingale, P.: A New Encoding of Alldifferent into SAT, *Proceedings of the 3rd International Workshop on Modelling and Reformulating Constraint Satisfaction Problems*, 2004.

[14] Griva, I., Nash, S. G., and Sofer, A.: *Linear and Nonlinear Optimization*, Society for Industrial Mathematics, 2009.

[15] Heule, M. and Szeider, S.: A SAT Approach to Clique-Width, *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, 2013, pp. 318–334.

[16] 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.

[17] Iwama, K. and Miyazaki, S.: SAT-Variable Complexity of Hard Combinatorial Problems, *Proceedings of the IFIP 13th World Computer Congress*, 1994, pp. 253–258.

[18] Lecoutre, C., Roussel, O., and van Dongen, M. R. C.: Promoting Robust Black-box Solvers Through Competitions, *Constraints*, Vol. 15, No. 3(2010), pp. 317–326.

[19] Metodi, A. and Codish, M.: Compiling finite domain constraints to SAT with BEE, *Theory and Practice of Logic Programming*, Vol. 12, No. 4-5(2012), pp. 465–483.

[20] Nabeshima, H., Iwanuma, K., and Inoue, K.: GlueMiniSat 2.2.8, *Proceedings of SAT Competition 2014*, 2014, pp. 35–36.

[21] 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.

[22] Rossi, F., van Beek, P., and Walsh, T.: *Handbook of Constraint Programming*, Elsevier, 2006.

[23] Soh, T., Inoue, K., Tamura, N., Banbara, M., and Nabeshima, H.: A SAT-based Method for Solving the Two-dimensional Strip Packing Problem, *Fundamenta Informaticae*, Vol. 102, No. 3-4(2010), pp. 467–487.

[24] Soh, T., Tamura, N., and Banbara, M.: Scarab: A Rapid Prototyping Tool for SAT-based Constraint Programming Systems, *Proceedings of the 16th International Conference on Theory and Applications of Satisfiability Testing (SAT 2013), LNCS 7962*, July 2013, pp. 429–436.

[25] Stojadinovic, M. and Maric, F.: meSAT: multiple encodings of CSP to SAT, *Constraints*, Vol. 19, No. 4(2014), pp. 380–403.

[26] Tamura, N., Banbara, M., and Soh, T.: PB-Sugar: Compiling Pseudo-Boolean Constraints to SAT with Order Encoding, *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2013), IEEE*, November 2013, pp. 1020–1027.

[27] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M.: Compiling Finite Linear CSP into SAT, *Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming (CP 2006), LNCS 4204*, 2006, pp. 590–603.

[28] Tamura, N., Taga, A., Kitagawa, S., and Banbara, M.: Compiling Finite Linear CSP into SAT, *Constraints*, Vol. 14, No. 2(2009), pp. 254–272.

[29] 田村直之, 丹生智也, 番原睦則: SAT 変換に基づく制約ソルバーとその性能評価, コンピュータソフトウェア, Vol. 27, No. 4(2010), pp. 183–196.

[30] Tseitin, G. S.: On the complexity of derivations in the propositional calculus, *Studies in Mathematics and Mathematical Logic Part II*, (1968), pp. 115–125.

[31] Walsh, T.: SAT v CSP, *Proceedings of the 6th International Conference on Principles and Practice of Constraint Programming (CP 2000)*, 2000, pp. 441–456.