数独問題におけるSAT符号化の影響の評価

鶴田　唯一[1]　楢崎　修二[2]

概要：近年，SAT ソルバと呼ばれる，制約充足問題を解くためのプログラムの性能が急速に向上してきている．そのため，身の回りの様々な問題を SAT 問題に変換し，簡単に，高速に処理するための研究が行われている．一般的な制約充足問題を命題論理式を用いた SAT 問題に変換する SAT 符号化は処理速度に影響を与える．グラフ彩色問題については，順序符号化法と呼ばれる手法が充足可能，不可能な両問題において平均的に優れた性能を持っているということがわかっている．しかし，その他の問題についても同様の結果になるかは不明である．そこで，本研究ではグラフ彩色問題によく似た性質を持つ数独問題に対して，各符号化法を施し，処理速度の比較を行うことで調査した．比較する符号化法は，充足可能なグラフ彩色問題について最も処理速度の速い多値符号化法，符号化法の中で出現する変数が最も少なくなる対数符号化法，順序関係を用いた順序符号化法の 3 つである．実験から，数独問題に対しては，順序符号化法よりも多値符号化法の方が充足可能，不可能の両問題において優れた性能を持っていることが判明した．よって，すべての問題に対して順序符号化法が優れているとは言えない．

キーワード：探索と推論，SAT，充足性問題, 知識表現

1. はじめに

近年，SAT ソルバと呼ばれる，制約充足問題を解くためのプログラムの性能が急速に向上してきている．そのため，身の回りの様々な問題を SAT 問題に変換し，簡単に，高速に処理するための研究が行われている．

一般的な制約充足問題を命題論理式を用いた SAT 問題に変換することを，「SAT 符号化」といい，様々な手法が提案されている．

グラフ彩色問題においての実験では，順序符号化法と呼ばれる手法が，充足可能，不可能な問題の両方において平均的に優れた結果を出している [1]．しかし，似たような別の問題についても同様になるかは不明である．そこで，本研究ではグラフ彩色問題に性質の似た数独問題に対して，充足可能なグラフ彩色問題について最も処理速度の速い多値符号化法，符号化法の中で出現する変数が最も少なくなる対数符号化法，順序関係を用いた順序符号化法を用いた場合のそれぞれの処理速度を調べた．

まず，SAT と SAT 符号化についてを詳しく説明する．次に，数独問題に対する SAT 符号化の手法とその実装についてを説明する．そこから，各符号化法による性能比較を行い，実験結果を基に考察を行う．

[1] 長崎大学, Nagasaki University
b609321@nagasaki-u.ac.jp
[2] 長崎大学工学研究科,Nagasaki University
narazaki@cs.cis.nagasaki-u.ac.jp

2. SAT と SAT 符号化

2.1 SAT とは

SAT(satisfiablility problem) とは，ある命題論理式に対して，それに含まれる真偽値をとる論理変数に真または偽を当てはめることによって，全体の値を真にできるか，という問題である．また，命題論理式の全体の値を真にできることを充足可能といい，できない場合を充足不能という．以下で，例題を交えてその解法を説明する．(図 1 参照)

x^1, x^2, x^3 という論理変数について，以下の命題論理式が与えられたとする．

$(x^1 \vee \neg x^3) \wedge (\neg x^1 \vee \neg x^2 \vee x^3) \wedge (\neg x^2 \vee x^3)$

はじめに，先頭の x^1 に真を割り当てると，x^1 を含む節は真とならなければならないので消すことができ，各節から $\neg x^1$ を消すことができるので，以下のようになる．

$(\neg x^2 \vee x^3) \wedge (\neg x^2 \vee x^3)$

さらに，先頭の x^2 に偽を割り当て，$\neg x^2$ を含む節を消し，x^2 を消すことができ，最終的に残った x^3 に真を入れることで，この命題論式を全体として真にすることができる．よって，この命題論理式は充足可能であるといえる．

SAT は NP 完全性が示された最初の問題である [11]．

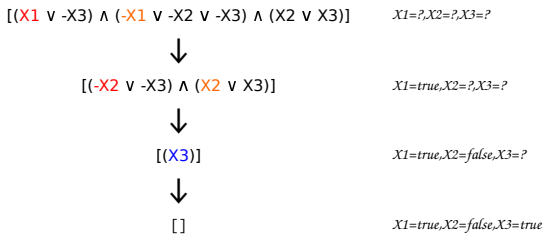


図 1 SAT 問題における解法の手順

そのため，計算機関係の研究の分野では，この SAT がとても重要な役割を担っている．ある問題が NP 完全であるかどうかについて考える場合に，SAT を利用することでその証明を行うことができるからである．

また，実生活においても，SAT は広く使われている．例えば，SAT を用いることで，グラフ彩色問題や巡回セールスマン問題，数独などのパズルゲームなどの問題が，充足可能であるかどうかを示し，解答可能であるかどうかを調べることに役立っている．

SAT を解くためのプログラムのことを SAT ソルバという．SAT ソルバに与える SAT は通常の場合，連言標準形 (以下，CNF) で記述される．CNF とは，∧(かつ)，∨(または)，¬(否定) の 3 つの論理式と任意の論理変数からなる命題論理式について，¬ を含む論理変数を ∨ で繋いだものを節として，複数の節を ∧ で繋いだもののことを言う．SAT ソルバは入力として CNF を受け取り，各論理変数に真偽値を割り当てていく．すべての論理変数に真偽値が割り当てられた場合を充足可能，そうでない場合を充足不能として，結果として出力する．この SAT ソルバの代表的なものとして，minisat が挙げられる [10]．

近年では，SAT ソルバの性能が向上し，高速に解を導くことができるようになった．そのため，スケジューリング問題や制約充足問題，制約最適化問題，定理証明などの幅広い問題を，SAT に変換する手法が研究されている．

ある問題を SAT に変換することを，SAT 符号化といい，SAT 符号化の方法には 2 つの分類が存在する．ひとつが「整数変数の符号化法」，もうひとつが「制約の符号化法」である [1]．

2.2 SAT 符号化の手法

SAT 符号化のうちの一つである整数変数の符号化法には 3 つの種類が存在する．

1 つ目が，**各整数変数** X と各整数定数 a に対して，$X = a$ を意味する命題変数を用いる方法である [2][3]．この方法を用いた主な符号化として，直接符号化法と多値符号化法，支持符号化法がある．直接符号化法は，at-least-one 節および at-most-one 節が必要となり，そこに制約の違反点集合を追加することで符号化がなされる．多値符号化法は，直接符号化法にあった at-most-one 節を省略したものである [4]．支持符号化法は，at-least-one 節および at-most-one 節が必要となり，そこに制約の支持点集合を追加することで符号化がなされる．

2 つ目が，各**整数変数** X の 2 進数表現に着目し，X の i 桁目が 1 に等しいことを意味する命題変数を用いる方法である [5][6]．この方法を用いた主な符号化として，対数符号化と対数支持符号化法がある．対数符号化法は，ドメインに含まれない値を除外する節が必要となり，そこに制約の違反点集合を追加することで符号化がなされる．対数支持符号化法は，ドメインに含まれない値を除外する節が必要となり，そこに制約の支持点集合を追加することで符号化がなされる．

3 つ目が，各**整数変数** X と各整数定数 a に対して，$X \leq a$ を意味する命題変数を用いる方法である [7][8]．この方法を用いた主な符号化として，順序符号化法がある．順序符号化法は，命題変数間の関係を表す節が必要となり，そこに違反する範囲を追加することで符号化がなされる．

それぞれの符号化法で，使用する命題変数の個数や作成される節数が異なる．例えば，それぞれの符号化法において，**整数変数** X の値域が 1 から 4 のときを考える．以下では，命題変数 x に 1(真) を割り当てる場合 x，0(偽) を割り当てる場合は $-x$ と表記する．

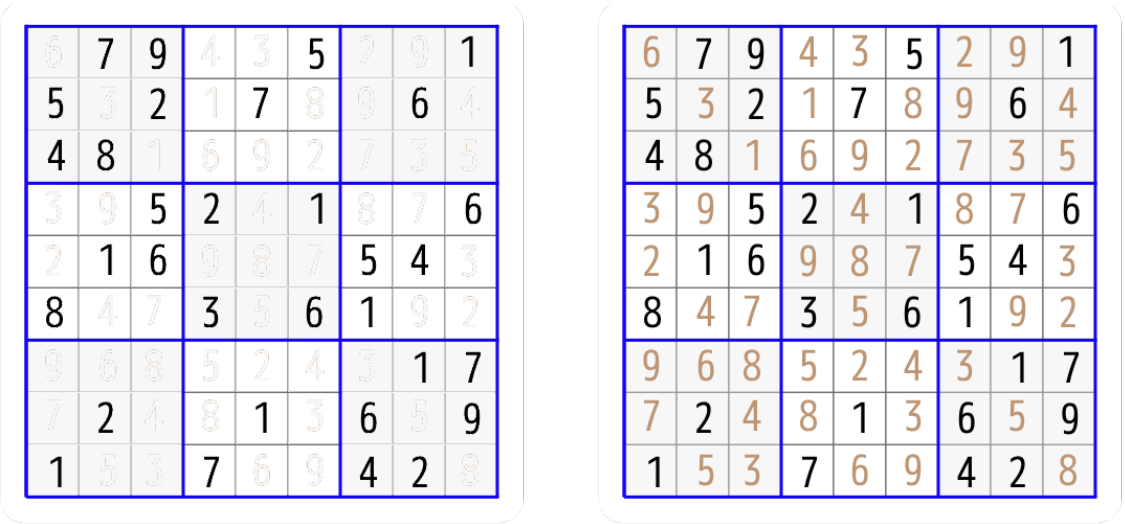
直接符号化法，多値符号化法では，**整数変数** X に対して，命題変数 x^1, x^2, x^3, x^4 を割り当てる．この x^1, x^2, x^3, x^4 は 1 から 4 の数に対応しており，X が 2 という値をとる場合，各命題変数は $-x^1, x^2, -x^3, -x^4$ と表される．

対数符号化法では，4 までの数は 2 進数だと 11 までを用いて 2 ビットで表現可能なので，**整数変数** X に対して，命題変数 x^1, x^2 を割り当てる．この x^1, x^2 はそれぞれの x^1 が 2 ビット目に,x^2 が 1 ビット目に対応しており，X が 2 という値をとる場合，各命題変数は $-x^1, x^2$ となる．

順序符号化法では，**整数変数** X に対して，命題変数 x^1, x^2, x^3 を割り当てる．この x^1, x^2, x^3 は，それぞれ $(X \leq 1), (X \leq 2), (X \leq 3)$ に対応している ($X \leq 4$ は常に真なので必要ない)．X が 2 という値をとる場合，各命題変数は $-x^1, x^2, x^3$ となる．

直接符号化法とその改良型である多値符号化法なら，節の数が少ない多値符号化法の方が処理にかかる時間が短くて済む．その具体例として，グラフ彩色問題と呼ばれる，与えられた無向有限グラフにおいて，隣り合う点同士が同一の色にならないように色分けし，その最小の色数を求める問題での性能比較がある [1]．

この実験によれば，直接，多値，支持，対数，対数支持，順序の 6 種類の符号化を用い，グラフ彩色問題を SAT 問題に符号化した問について平均 CPU 時間を計測した場合，符号化の手法によって処理時間に差が生じることが示されている．また，順序符号化法が彩色可能な場合も



与えられる問題　　解答例

図 2　数独問題の一例

彩色不能な場合も平均的に良い性能を示している [1].

3. 数独問題に対する SAT 符号化

3.1 数独とは

グラフ彩色問題については以上のような実験結果が出ているが，その他の問題に対して各符号化を行った場合はどうなるかを調べていく．本研究では，グラフ彩色問題とよく似た性質を持つ**数独**と呼ばれるパズルゲームについて，処理速度の比較を行う．この数独とは，以下のルールに従って，1 から 9 までの数を 9×9 の 81 マスに当てはめていくパズルゲームのことである．(図 2 参照)

- ルール 1:1 つのマスには 1 から 9 の数が 1 つだけ入る
- ルール 2:縦 1 列（計 9 マス）には，1 から 9 の数が 1 つずつ入る
- ルール 3:横 1 列（計 9 マス）には，1 から 9 の数が 1 つずつ入る
- ルール 4:3×3 で囲まれた 9 マスには，1 から 9 の数が 1 つずつ入る

基本的には，以上 4 つのルールに従って問題を解いていくが，派生系として「対角線上に 1 から 9 の数が 1 つずつ入る」というルールを持ったもの，1 から 16 までの数を 16×16 の 256 マスに当てはめていくものなどがある．

3.2 探索プログラムによる数独の解法

一般的に，数独は以下のような手順で解を導いてゆく．

(1) 数の入っていないマスに，縦・横・3×3 の大きなマスの中を見て候補を入れていく．

(2) 先頭のマスから見ていき，候補が一つしか入っていないマスがあれば，その値を入れる．値が入ったら，値を入れたマスの縦・横・3×3 のマスをチェックし，入れた数が候補にあれば消す．

(3) 2 を繰り返し，候補が一つのマスがなくなったら，先頭のマスから縦・横・3×3 のマスを見ていき，候補が複数入っているマスの中から，候補が 1 つだけ入るマスを見つけ，その数を値として入れる．値を入れたら，値を入れたマスの縦・横・3×3 のマスをチェックし，入れた数が候補にあれば消す．

(4) 2 と 3 を繰り返し，すべてのマスに値が入ったら，それが解となる．

3.3 多値符号化法

ここからは，9×9 の数独問題を SAT ソルバで解ける形に変換することを考える．実装する符号化法は，充足可能なグラフ彩色問題について最も処理速度の速い多値符号化法，符号化法の中で出現する変数が最も少なくなる対数符号化法，順序関係を用いた順序符号化法の計 3 つ．また以下からは，一行一列のマスを $X_{1,1}$，一行二列のマスを $X_{1,2}$，$\cdots$，九行九列のマスを $X_{9,9}$ とする．

多値符号化法を用いる場合，各整数変数 $X_{i,j}$ の取り得る値が 9 個あるので，$X_{1,1}$ に $x^1, x^2, x^3, \cdots, x^9$，$X_{1,2}$ に $x^{10}, x^{11}, x^{12}, \cdots, x^{18}$ という風に各マスに 9 個，計 729 変数を割り当てる．(図 3 参照) 例えば，$X_{1,1}$ に 3 が入る場合，変数の組み合わせは $-x^1, -x^2, x^3, -x^4, -x^5, -x^6, -x^7, -x^8, -x^9$ となる．この変数を元に各ルールを符号化していく．

まず，ルール 1 について，各マスに割り当てた 9 個の変数のうち 1 つだけが真となるような論理式を作ればよい．これは言い換えると，「各マスに割り当てた 9 個の変数のうち 2 つ以上が真となる」の否定である．よって，$X_{1,1}$ について言えば以下のような節になる．

$\neg(x^1 \wedge x^2 \wedge \neg x^3 \wedge \neg x^4 \wedge \neg x^5 \wedge \neg x^6 \wedge \neg x^7 \wedge \neg x^8 \wedge \neg x^9) \vee \neg(x^1 \wedge \neg x^2 \wedge x^3 \wedge \neg x^4 \wedge \neg x^5 \wedge \neg x^6 \wedge \neg x^7 \wedge \neg x^8 \wedge$

$x^1,-x^2,-x^3,-x^4,-x^5,-x^6,-x^7,-x^8,-x^9(X_{1,1}=1)$	$x^{10},-x^{11},-x^{12},-x^{13},-x^{14},-x^{15},-x^{16},-x^{17},-x^{18}(X_{1,2}=1)$	
$-x^1,x^2,-x^3,-x^4,-x^5,-x^6,-x^7,-x^8,-x^9(X_{1,1}=2)$	$-x^{10},x^{11},-x^{12},-x^{13},-x^{14},-x^{15},-x^{16},-x^{17},-x^{18}(X_{1,2}=2)$	
$-x^1,-x^2,x^3,-x^4,-x^5,-x^6,-x^7,-x^8,-x^9(X_{1,1}=3)$	$-x^{10},-x^{11},x^{12},-x^{13},-x^{14},-x^{15},-x^{16},-x^{17},-x^{18}(X_{1,2}=3)$	
$-x^1,-x^2,-x^3,x^4,-x^5,-x^6,-x^7,-x^8,-x^9(X_{1,1}=4)$	$-x^{10},-x^{11},-x^{12},x^{13},-x^{14},-x^{15},-x^{16},-x^{17},-x^{18}(X_{1,2}=4)$	
$-x^1,-x^2,-x^3,-x^4,x^5,-x^6,-x^7,-x^8,-x^9(X_{1,1}=5)$	$-x^{10},-x^{11},-x^{12},-x^{13},x^{14},-x^{15},-x^{16},-x^{17},-x^{18}(X_{1,2}=5)$	$\cdots$
$-x^1,-x^2,-x^3,-x^4,-x^5,x^6,-x^7,-x^8,-x^9(X_{1,1}=6)$	$-x^{10},-x^{11},-x^{12},-x^{13},-x^{14},x^{15},-x^{16},-x^{17},-x^{18}(X_{1,2}=6)$	
$-x^1,-x^2,-x^3,-x^4,-x^5,-x^6,x^7,-x^8,-x^9(X_{1,1}=7)$	$-x^{10},-x^{11},-x^{12},-x^{13},-x^{14},-x^{15},x^{16},-x^{17},-x^{18}(X_{1,2}=7)$	
$-x^1,-x^2,-x^3,-x^4,-x^5,-x^6,-x^7,x^8,-x^9(X_{1,1}=8)$	$-x^{10},-x^{11},-x^{12},-x^{13},-x^{14},-x^{15},-x^{16},x^{17},-x^{18}(X_{1,2}=8)$	
$-x^1,-x^2,-x^3,-x^4,-x^5,-x^6,-x^7,-x^8,x^9(X_{1,1}=9)$	$-x^{10},-x^{11},-x^{12},-x^{13},-x^{14},-x^{15},-x^{16},-x^{17},x^{18}(X_{1,2}=9)$	
$x^{82},-x^{83},-x^{84},-x^{85},-x^{86},-x^{87},-x^{88},-x^{89},-x^{90}(X_{2,1}=1)$	$x^{91},-x^{92},-x^{93},-x^{94},-x^{95},-x^{96},-x^{97},-x^{98},-x^{99}(X_{2,2}=1)$	
$-x^{82},x^{83},-x^{84},-x^{85},-x^{86},-x^{87},-x^{88},-x^{89},-x^{90}(X_{2,1}=2)$	$-x^{91},x^{92},-x^{93},-x^{94},-x^{95},-x^{96},-x^{97},-x^{98},-x^{99}(X_{2,2}=2)$	
$-x^{82},-x^{83},x^{84},-x^{85},-x^{86},-x^{87},-x^{88},-x^{89},-x^{90}(X_{2,1}=3)$	$-x^{91},-x^{92},x^{93},-x^{94},-x^{95},-x^{96},-x^{97},-x^{98},-x^{99}(X_{2,2}=3)$	
$-x^{82},-x^{83},-x^{84},x^{85},-x^{86},-x^{87},-x^{88},-x^{89},-x^{90}(X_{2,1}=4)$	$-x^{91},-x^{92},-x^{93},x^{94},-x^{95},-x^{96},-x^{97},-x^{98},-x^{99}(X_{2,2}=4)$	
$-x^{82},-x^{83},-x^{84},-x^{85},x^{86},-x^{87},-x^{88},-x^{89},-x^{90}(X_{2,1}=5)$	$-x^{91},-x^{92},-x^{93},-x^{94},x^{95},-x^{96},-x^{97},-x^{98},-x^{99}(X_{2,2}=5)$	$\cdots$
$-x^{82},-x^{83},-x^{84},-x^{85},-x^{86},x^{87},-x^{88},-x^{89},-x^{90}(X_{2,1}=6)$	$-x^{91},-x^{92},-x^{93},-x^{94},-x^{95},x^{96},-x^{97},-x^{98},-x^{99}(X_{2,2}=6)$	
$-x^{82},-x^{83},-x^{84},-x^{85},-x^{86},-x^{87},x^{88},-x^{89},-x^{90}(X_{2,1}=7)$	$-x^{91},-x^{92},-x^{93},-x^{94},-x^{95},-x^{96},x^{97},-x^{98},-x^{99}(X_{2,2}=7)$	
$-x^{82},-x^{83},-x^{84},-x^{85},-x^{86},-x^{87},-x^{88},x^{89},-x^{90}(X_{2,1}=8)$	$-x^{91},-x^{92},-x^{93},-x^{94},-x^{95},-x^{96},-x^{97},x^{98},-x^{99}(X_{2,2}=8)$	
$-x^{82},-x^{83},-x^{84},-x^{85},-x^{86},-x^{87},-x^{88},-x^{89},x^{90}(X_{2,1}=9)$	$-x^{91},-x^{92},-x^{93},-x^{94},-x^{95},-x^{96},-x^{97},-x^{98},x^{99}(X_{2,2}=9)$	
$\cdots$	$\cdots$	$\cdots$

図 3 多値符号化法における変数の割り当て

$\neg x^9) \vee \neg(x^1 \wedge \neg x^2 \wedge \neg x^3 \wedge x^4 \wedge \neg x^5 \wedge \neg x^6 \wedge \neg x^7 \wedge \neg x^8 \wedge \neg x^9) \cdots \vee \neg(x^1 \wedge x^2 \wedge x^3 \wedge x^4 \wedge x^5 \wedge x^6 \wedge x^7 \wedge x^8 \wedge x^9)$

このように 246 節必要になるのだが，これらは多値符号化法において，「割り当てた変数のうち少なくとも 1 つが真となる」という以下の節によって，すべて充足することが可能である [4].

$x^1 \vee x^2 \vee x^3 \vee x^4 \vee x^5 \vee x^6 \vee x^7 \vee x^8 \vee x^9$

よって，このような 1 節が各マスごとに存在するので，多値符号化法においてルール 1 は 81 節に符号化される．

次に，ルール 2 について，1 行目の 1 という数について考える．「縦に並ぶマスには 1 という数が 1 つしか入らない」という条件は，言い換えると「縦に並ぶマス同士に 1 という数が入る」の否定である．従って，以下のような節になる．

$\neg(x^1 \wedge x^{82}) \wedge \neg(x^1 \wedge x^{163}) \wedge \cdots \wedge \neg(x^1 \wedge x^{649}) \wedge \neg(x^{82} \wedge x^{163}) \wedge \cdots \wedge \neg(x^{568} \wedge x^{649})$

これを変形したものが以下である．

$(\neg x^1 \vee \neg x^{82}) \wedge (\neg x^1 \vee \neg x^{163}) \wedge \cdots \wedge (\neg x^1 \vee \neg x^{649}) \wedge (\neg x^{82} \vee \neg x^{163}) \wedge \cdots \wedge (\neg x^{568} \vee \neg x^{649})$

この 36 節が 1 から 9 までの 9 個の各数，各行ごとに存在するので，ルール 2 は 2916 節に符号化される．

ルール 3 について，1 列目の 1 という数について考える．「横に並ぶマスには 1 という数が 1 つしか入らない」という条件は，言い換えると「横に並ぶマス同士に 1 という数が入る」の否定である．従って，以下のような節になる．

$\neg(x^1 \wedge x^{10}) \wedge \neg(x^1 \wedge x^{19}) \wedge \cdots \wedge \neg(x^1 \wedge x^{73}) \wedge \neg(x^{10} \wedge x^{19}) \wedge \cdots \wedge \neg(x^{64} \wedge x^{73})$

これを変形したものが以下である．

$(\neg x^1 \vee \neg x^{10}) \wedge (\neg x^1 \vee \neg x^{19}) \wedge \cdots \wedge (\neg x^1 \vee \neg x^{73}) \wedge (\neg x^{10} \vee \neg x^{19}) \wedge \cdots \wedge (\neg x^{64} \vee \neg x^{73})$

この 36 節が 1 から 9 までの 9 個の各数，各列ごとに存在するので，ルール 3 は 2916 節に符号化される．

最後に，ルール 4 について，先頭の 3×3 のマスの 1 という数について考える．「3×3 のマスの中には 1 という数が 1 つしか入らない」という条件は，言い換えると「3×3 のマスの中のマス同士に 1 という数が入る」の否定である．従って，以下のような節になる．

$\neg(x^1 \wedge x^{10}) \wedge \neg(x^1 \wedge x^{19}) \wedge \neg(x^1 \wedge x^{82}) \wedge \neg(x^1 \wedge x^{91}) \wedge \neg(x^1 \wedge x^{100}) \wedge \neg(x^1 \wedge x^{163}) \wedge \neg(x^1 \wedge x^{172}) \wedge \neg(x^1 \wedge x^{181}) \wedge \neg(x^{10} \wedge x^{19}) \wedge \neg(x^{10} \wedge x^{82}) \wedge \cdots \wedge \neg(x^{172} \wedge x^{181})$

これを変形したものが以下である．

$(\neg x^1 \vee \neg x^{10}) \wedge (\neg x^1 \vee \neg x^{19}) \wedge (\neg x^1 \vee \neg x^{82}) \wedge (\neg x^1 \vee \neg x^{91}) \wedge (\neg x^1 \vee \neg x^{100}) \wedge (\neg x^1 \vee \neg x^{163}) \wedge (\neg x^1 \vee \neg x^{172}) \wedge (\neg x^1 \vee \neg x^{181}) \wedge (\neg x^{10} \vee \neg x^{19}) \wedge (\neg x^{10} \vee \neg x^{82}) \wedge \cdots \wedge (\neg x^{172} \vee \neg x^{181})$

この 36 節が 1 から 9 までの 9 個の各数，各 3×3 のマスごとに存在するので，ルール 4 は 2916 節に符号化される．

よって，9×9 の数独問題に対して多値符号化法を行った場合，729 変数，8829 節になる．

3.4 対数符号化法

対数符号化法を用いる場合，$X_{1,1}$ に x^1, x^2, x^3, x^4，$X_{1,2}$ に x^5, x^6, x^7, x^8 という風に各マスに 4 個，計 324 変数を割り当てる．例えば，$X_{1,1}$ に 3 が入る場合，変数の組み合わせは $-x^1, -x^2, x^3, -x^4$ となる．この変数を元に各ルールを符号化していく．

まず，ルール 1 を多値符号化法と同じようにして，$X_{1,1}$ について以下のような節に符号化する．

$\neg(x^1 \wedge \neg x^2 \wedge \neg x^3 \wedge x^4) \wedge \neg(x^1 \wedge \neg x^2 \wedge x^3 \wedge \neg x^4) \wedge \neg(x^1 \wedge \neg x^2 \wedge x^3 \wedge x^4) \wedge \neg(x^1 \wedge x^2 \wedge \neg x^3 \wedge \neg x^4) \wedge \neg(x^1 \wedge x^2 \wedge \neg x^3 \wedge x^4) \wedge \neg(x^1 \wedge x^2 \wedge x^3 \wedge \neg x^4) \wedge \neg(x^1 \wedge x^2 \wedge x^3 \wedge x^4)$

これを変形したものが以下である．

$(\neg x^1 \vee x^2 \vee x^3 \vee \neg x^4) \wedge (\neg x^1 \vee x^2 \vee \neg x^3 \vee x^4) \wedge (\neg x^1 \vee x^2 \vee \neg x^3 \vee \neg x^4) \wedge (\neg x^1 \vee \neg x^2 \vee x^3 \vee x^4) \wedge (\neg x^1 \vee \neg x^2 \vee$

$x^3 \vee \neg x^4) \wedge (\neg x^1 \vee \neg x^2 \vee \neg x^3 \vee x^4) \wedge (\neg x^1 \vee \neg x^2 \vee \neg x^3 \vee \neg x^4)$

この 7 節が各マスごとに存在するので，対数符号化法においてルール 1 は 567 節に符号化される．

次に，ルール 2 についても，多値符号化法と同様にして，以下のような節になる．

$\neg(\neg x^1 \wedge \neg x^2 \wedge \neg x^3 \wedge \neg x^4 \wedge \neg x^{37} \wedge \neg x^{38} \wedge \neg x^{39} \wedge \neg x^{40}) \wedge \neg(\neg x^1 \wedge \neg x^2 \wedge \neg x^3 \wedge \neg x^4 \wedge \neg x^{73} \wedge \neg x^{74} \wedge \neg x^{75} \wedge \neg x^{76}) \wedge \cdots \wedge \neg(\neg x^1 \wedge \neg x^2 \wedge \neg x^3 \wedge \neg x^4 \wedge \neg x^{289} \wedge \neg x^{290} \wedge \neg x^{291} \wedge \neg x^{292}) \wedge \neg(\neg x^{37} \wedge \neg x^{38} \wedge \neg x^{39} \wedge \neg x^{40} \wedge \neg x^{73} \wedge \neg x^{74} \wedge \neg x^{75} \wedge \neg x^{76}) \wedge \cdots \wedge \neg(\neg x^{253} \wedge \neg x^{254} \wedge \neg x^{255} \wedge \neg x^{256} \wedge \neg x^{289} \wedge \neg x^{290} \wedge \neg x^{291} \wedge \neg x^{292})$

これを変形したものが以下である．

$(x^1 \vee x^2 \vee x^3 \vee x^4 \vee x^{37} \vee x^{38} \vee x^{39} \vee x^{40}) \wedge (x^1 \vee x^2 \vee x^3 \vee x^4 \vee x^{73} \vee x^{74} \vee x^{75} \vee x^{76}) \wedge \cdots \wedge (x^1 \vee x^2 \vee x^3 \vee x^4 \vee x^{289} \vee x^{290} \vee x^{291} \vee x^{292}) \wedge (x^{37} \vee x^{38} \vee x^{39} \vee x^{40} \vee x^{73} \vee x^{74} \vee x^{75} \vee x^{76}) \wedge \cdots \wedge (x^{253} \vee x^{254} \vee x^{255} \vee x^{256} \vee x^{289} \vee x^{290} \vee x^{291} \vee x^{292})$

この 36 節が 1 から 9 までの 9 個の各数，各行ごとに存在するので，ルール 2 は 2916 節に符号化される．

ルール 3 については，ルール 2 と同様にして，36 節が 1 から 9 までの 9 個の各数，各列ごとに存在するので，ルール 3 は 2916 節に符号化される．

ルール 4 についても，ルール 2 と同様にして 36 節が 1 から 9 までの 9 個の各数，各 3×3 のマスごとに存在するので，ルール 4 は 2916 節に符号化される．

よって，9×9 の数独問題に対して対数符号化法を行った場合，324 変数，9315 節になる．

3.5 順序符号化法

順序符号化法を用いる場合，$X_{1,1}$ に $x^1, x^2, x^3, \cdots, x^8$，$X_{1,2}$ に $x^9, x^{10}, x^{11}, \cdots, x^{16}$ という風に各マスに 8 個，計 648 変数を割り当てる．例えば，$X_{1,1}$ に 3 が入る場合，変数の組み合わせは $-x^1, -x^2, x^3, x^4, x^5, x^6, x^7, x^8$ となる．この変数を元に各ルールを符号化していく．

まず，ルール 1 について，$X_{1,1}$ について言えば割り当てた 8 個の変数を用いて，以下のような節を追加する．

$(\neg x^1 \vee x^2) \wedge (\neg x^2 \vee x^3) \wedge (\neg x^3 \vee x^4) \wedge (\neg x^4 \vee x^5) \wedge (\neg x^5 \vee x^6) \wedge (\neg x^6 \vee x^7) \wedge (\neg x^7 \vee x^8)$

この 7 節が各マスごとに存在するので，順序符号化法においてルール 1 は 567 節に符号化される．

次に，ルール 2 について，縦に並んだ 2 つのマス $X_{1,1}$,$X_{2,1}$ について考える．$X_{1,1}$ と $X_{2,1}$ にそれぞれ同じ数が入らないという制約は，言い換えると $(X_{1,1} - X_{2,1} \leq -1) \vee (X_{2,1} - X_{1,1} \leq -1)$ と表すことができる．これを Tseitin 変換 [9] を用いて，$(p \vee q) \wedge [\neg p \vee (X_{1,1} - X_{2,1} \leq -1)] \wedge [\neg q \vee (X_{2,1} - X_{1,1} \leq -1)]$ と変換してから符号化を行う．(p,q は新しい命題変数) さらに，p と q は排他的であるので，q を $\neg p$ で置き換えることができる．したがって，この制約は以下のような節に符号化できる．

$(\neg p \vee \neg x^{73}) \wedge (\neg p \vee x^1 \vee \neg x^{74}) \wedge (\neg p \vee x^2 \vee \neg x^{75}) \wedge (\neg p \vee x^3 \vee \neg x^{76}) \wedge (\neg p \vee x^4 \vee \neg x^{77}) \wedge (\neg p \vee x^5 \vee \neg x^{78}) \wedge (\neg p \vee x^6 \vee \neg x^{79}) \wedge (\neg p \vee x^7 \vee \neg x^{80}) \wedge (\neg p \vee x^8) \wedge (p \vee \neg x^1) \wedge (p \vee x^{73} \vee \neg x^2) \wedge (p \vee x^{74} \vee \neg x^3) \wedge (p \vee x^{75} \vee \neg x^4) \wedge (p \vee x^{76} \vee \neg x^5) \wedge (p \vee x^{77} \vee \neg x^6) \wedge (p \vee x^{78} \vee \neg x^7) \wedge (p \vee x^{79} \vee \neg x^8) \wedge (p \vee x^{80})$

この 18 節が各行ごとに 36 通り組み合わせが存在するので，ルール 2 は 5832 節に符号化される．また，変数 p は組み合わせごとに新たに 324 個の変数が必要になる．

ルール 3 についても，ルール 2 と同様にして横に並んだ 2 つのマス $X_{1,1}$,$X_{1,2}$ について考えると，以下のような節に符号化できる．

$(\neg p \vee \neg x^9) \wedge (\neg p \vee x^1 \vee \neg x^{10}) \wedge (\neg p \vee x^2 \vee \neg x^{11}) \wedge (\neg p \vee x^3 \vee \neg x^{12}) \wedge (\neg p \vee x^4 \vee \neg x^{13}) \wedge (\neg p \vee x^5 \vee \neg x^{14}) \wedge (\neg p \vee x^6 \vee \neg x^{15}) \wedge (\neg p \vee x^7 \vee \neg x^{16}) \wedge (\neg p \vee x^8) \wedge (p \vee \neg x^1) \wedge (p \vee x^9 \vee \neg x^2) \wedge (p \vee x^{10} \vee \neg x^3) \wedge (p \vee x^{11} \vee \neg x^4) \wedge (p \vee x^{12} \vee \neg x^5) \wedge (p \vee x^{13} \vee \neg x^6) \wedge (p \vee x^{14} \vee \neg x^7) \wedge (p \vee x^{15} \vee \neg x^8) \wedge (p \vee x^{16})$

この 18 節が各列ごとに 36 通り組み合わせが存在するので，ルール 3 は 5832 節に符号化される．また，変数 p は組み合わせごとに新たに 324 個の変数が必要になる．

ルール 4 についても同様にして，3×3 のマスの中で 36 通りの組み合わせが存在し，それが 9 個あるので，5832 節に符号化され，新たに 324 個の変数が必要になる．

よって，9×9 の数独問題に対して順序符号化法を行った場合，1620 変数，18063 節になる．

4. 各 SAT 符号化法の比較

ここまでで，数独問題に対する SAT 符号化の手法と実装を説明したが，それらが処理速度にどれ程の影響を及ぼすのかを実際に調べてみることにする．その結果から，最も数独問題に適した符号化法と各符号化法の特性についてを調査していく．

4.1 使用する SAT ソルバと実験環境

今回使用する SAT ソルバは以下の 3 つである．1 つ目に, minisat と呼ばれる SAT ソルバの中でも特に優れた性能で知られた SAT ソルバを用いた [10]．2 つ目が, funsat と呼ばれる minisat を模してプログラミング言語の 1 つである Haskell を用いて記述された SAT ソルバである [12]．そして，我々が Haskell を用いて開発中の hSAT という SAT ソルバである．これらを用いて，各符号化について処理速度の比較を行った．

実験環境は Intel Core i7 $2.67GHz \times 8$, メモリ $5.8GByte$ の計算機上で行った．

4.2 比較する問題

比較には，一意に解が決まる問題，一意に解が決まらない問題，現実にはありえないが解がない (UNSAT とな

表 1 minisat を用いた平均処理速度の比較

符号化の種類	一意に解が決まる問題 (ms)	一意に解が決まらない問題 (ms)	解がない問題 (ms)
多値	11.247	10.619	10.255
対数	22.238	19.845	16.218
順序	16.918	14.945	14.286

表 2 funsat を用いた処理速度の比較

符号化の種類	一意に解が決まる問題 (s)	一意に解が決まらない問題 (s)	解がない問題 (s)
多値	0.172	0.140	0.168
対数	5.914	0.917	0.696
順序	1.175	0.388	0.450

表 3 hSAT を用いた処理速度の比較

符号化の種類	一意に解が決まる問題 (s)	一意に解が決まらない問題 (s)	解がない問題 (s)
多値	2.914	0.353	0.146
対数	-	-	-
順序	-	-	-

る) 問題を 10 問ずつ用いる．以上，計 30 問について 3 種類の符号化法で処理速度の計測を行い，各種の問題について平均処理速度を求め比較した．

4.3 minisat による比較

まず，minisat を用いた実験結果が表 1 である．この表を見る限り，多値符号化法がどのような問題に対しても最も処理速度が速いということがわかる．また，対数符号化法は問題によって処理速度にバラつきがあり，不安定であることがわかった．

4.4 funsat による比較

次に，funsat を用いた実験結果が表 2 である．この表からも，多値符号化法がどのような問題に対しても最も処理速度が速いということがわかる．特に，一意に解が決まる一般的な数独問題に対して言えば，その性能の良さは明らかである．

4.5 hSAT による比較

次に，hSAT を用いた実験結果が表 3 である．表中の'-'マークは 1 問につき 20 回の計測を行った場合，解答に 30 分以上かかり，計測を中断したため，平均値が算出できなかったことを意味する．この表から，多値符号化法以外の方法は 1 問計測するのにも 30 分以上時間がかかり，実用的でないということがわかった．

5. 考察

ここまでの実験の結果から，以下のようなことがわかった．

まず，一般的に知られている 9×9 の数独問題に対しては，多値符号化が最も処理速度が速く優れている．逆に，最も変数の数が少ない対数符号化は平均的に見れば，3 つの符号化の中で最も処理速度が遅いことがわかる．順序符号化に至っては，扱う変数の数が 1620 個と，多値符号化の 2 倍以上であるにも関わらず対数符号化よりも優れた性能を示している．よって，数独問題に対してもグラフ彩色問題と同様に，扱う変数の数が少ないからといって，処理速度が向上するわけではないということが示された．

次に，各符号化の節の数を見てみると，対数符号化が 9315 節であるのに対し，順序符号化は 18063 節も要している．それにも関わらず，順序符号化の方が対数符号化よりもよい結果が出ている．ここから，多値符号化には劣るものの，節数が他の符号化よりも多い順序符号化も，優秀な符号化法のひとつであるということが言える．

さらに，使用する SAT ソルバの種類によっても，その性能に大きな差が現れる．一意に解が決まる問題について言えば，minisat では多値符号化と対数符号化で約 2 倍の性能差なのに対して，funsat では約 34 倍も性能に差が生じている．

以上のように，すでに行われているグラフ彩色問題においては，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していたが，数独問題に関しては，多値符号化法が充足可能，不可能な問題の両方に対して最も優れた性能を示している．

これらの考察から，9×9 の数独問題に対して，最も優れた符号化法は多値符号化法であるということが言える．

6. おわりに

この実験を通して，数独問題に対する SAT 符号化の手法によって，処理速度に大きな影響を及ぼすという事が示された．結果，グラフ彩色問題とは異なり，数独問題に最も適した符号化法は**多値符号化法**であることが判明した．よって，グラフ彩色問題に似た性質を持つ問題であっても，順序符号化法を行うことが最も優れた方法ではないと言える．今回は 9×9 の数独問題において，このような結果が得られた．今後は，16×16 の数独問題や対角線に関する制約を追加した数独問題など調査の範囲を広げ，今回の結果が正しいことを確認したい．

参考文献

[1] 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).
[2] de Kleer,J.:A Comparison of ATMS and CSP techniques, Proc. 11th Int. Joint Conf. on Artificial Intelligence(IJCAI 89), pp.290-296 (1989).
[3] Walsh,T.:SAT v CSP, Proc. 6th Int. Conf. on Principles and Practice of Constraint Programming (CP 2000),pp.441-456 (2000).
[4] Selman,B.,Levesque,H.J. and Mitchell,D.G.: A new method for solving hard satisfiability problems, Proc.

10th National Conf. on Artificial Intelligence (AAAI-92), pp.440-446 (1992).
[5] Gelder,A.V.:Another look at graph coloring via propositional satisfiability,Discrete Applied Mathematics, Vol.156,No.2,pp.230-243 (2008).
[6] Iwama,K. and Miyazaki,S.:SAT-variable complexity of hard combinatorial problems, Proc.IFIP 13th World Computer Congress,pp.253-258 (1994).
[7] Tamura,N.,Taga,A.,Kitagawa,S. and Banbara,M.:Compiling finite linear CSP into SAT, Proc. 12th INt. Conf. on Principles and Practice of Constraint Programming (CP 2006), LNCS 4204,pp.590-603 (2006).
[8] Tamura,N.,Taga,A.,Kitagawa,S. and Banbara,M.:Compiling finite linear CSP into SAT,Constraints,Vol.14,No.2,pp.254-272 (2009).
[9] Prestwich,S.:Handbook of Satisdiability,chapter 12:CNF encodings,pp.75-97, IOS Press (2009).
[10] Een,N. and Sorensson,N:An extensible SAT-solver, Proc. 6th Int. Conf. on Theory and Applicatonsof Satisfiability Testing (SAT2003), pp.502-518 (2003).
[11] http://metabolomics.jp/wiki/Aritalab:Lecture/Algorithm/CooksTheorem
[12] Denis,B.: http://hackage.haskell.org/package/funsat