Vol.2015-SLDM-173 No.17
2015/12/2

# ナンバーリンク問題に対する命題論理式のエンコーディング法の評価について

松永 裕介[1,a)]

**概要**：本稿では SAT ソルバを用いてナンバーリンク問題を解く場合の問題のエンコーディング方式について数種類の方法を考案し，実験により評価を行った．その結果，各々のマス目に対してラベル値を表す変数を one-hot 符号化の形で割り当てる方式が最も効率的であることがわかった．

**キーワード**：ナンバーリンク, SAT

# On evaluation of encoding scheme of propsitional formula for number-link problem

MATSUNAGA YUSUKE[1,a)]

***Abstract:*** This paper presents comparison and evaluation results of encoding scheme of propositional formula for number-link problem. The experiments show that the scheme which introduces variables representing label number for each grid in one-hot encoding style is most efficient.

***Keywords:*** number link, SAT

## 1. はじめに

DA シンポジウム 2014 および 2015 ではアルゴリズムデザインコンテストと題してナンバーリンク問題を解くアルゴリズムのコンテストを行っている．この問題に対しては SAT ソルバを用いた手法 [3] がよい成績を残している．ただ，一口に SAT ソルバを用いるといっても，ナンバーリンク問題を命題論理式に変換するやり方はさまざまなものが考えられる．そこで，本稿ではいくつかのエンコーディング方式を考案し，実験を行ってその効率を評価した．また，外周に沿って経路を確定するヒューリスティックについても提案している．

本稿の構成は以下のようになっている．まず 2 節でナンバーリンク問題の定式化を行い，3 節でさまざまなエンコーディング手法を紹介する．4 節で外周に基づいてヒューリスティックについて紹介し，5 節で実験結果を示す．最後に 6 でまとめを行う．

## 2. ナンバーリンク問題

ナンバーリンク問題とはマス目で区切られた矩形中の同じ数字同士を他の線分と交わることのないように結ぶパズル問題である．もう少し厳密には，一つのマス目には高々一つの線分しか引くことができない．また，線はマス目上で直角に曲がることはできるが，斜めに引くことはできない，という制約のもとで同じ番号同士を結ぶ線分を求める問題である．この問題はグラフ理論の用語を用いると以下のように定式化できる．

- マス目をグラフの節点に対応させる．
- 隣接するマス目に対応する節点の間に枝を設ける．
- 一部の節点は 1〜$K$ のラベルを持つ．
- 同じ値を持つ節点は 2 つ存在する．
- 同じ値を持つ節点間を結ぶ径路を求める．ただし，異

[1] 九州大学大学院システム情報科学研究院
〒 819–0395 福岡市西区元岡 744
[a)] matsunaga@ait.kyushu-u.ac.jp

なる径路はおなじ節点を共有することはできない．
一般によく知られたナンバーリンク問題の多くは，すべてのマス目がいずれかの経路に使われているが，このことはナンバーリンクのルールではない．実際，DA シンポジウム 2015 のアルゴリズムデザインコンテストにおいては，未使用のマス目が多く現れる問題が出されており，解の評価基準に径路長と線分の曲がり回数が含まれている．

図 1(a) にナンバーリンク問題の例を示す．図 1(b) に対応するグラフ表現を示す．以降，ラベルを持つ節点を**終**端節点，それ以外の節点を**中**間節点と呼ぶことにする．

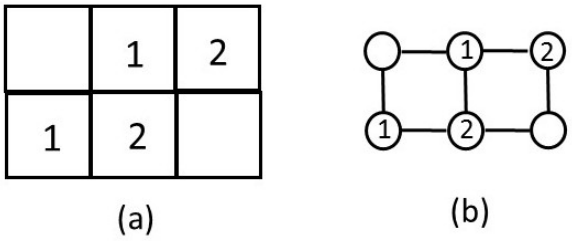


**図 1** ナンバーリンク問題

## 3. SAT 問題へのエンコーディング方式

[3] ではこのナンバーリンク問題を制約充足問題として定式化し，それを命題論理式に変換することで SAT ソルバを用いて解いている．本稿でも同様にナンバーリンク問題を命題論理式へ変換するアプローチをとっている．ただし，この命題論理式への変換 (符号化・エンコーディング) 方式は唯一ではない．そこで，いくつかのエンコーディング方式を考案し，実験によりその効率の評価を行った．

### 3.1 方式 1: 枝に 2 値変数，節点に多値変数を割り当てる方式

これは大まかには [3] と同様の方法である [*1]．解の径路上にある枝に対応した変数の値を 1 にして，そうでない場合を 0 で表し，節点に対応した変数にはその節点を通る径路のラベル値をもたせる，という方式である．この方式では枝と節点の変数に関して以下の制約を考える必要がある．以降，枝 $e$ に対応した変数も同じ記号 $e$ で表すものとする．節点 $v$ とそれに対応した変数に関しても同様とする．

- 節点 $v$ が終端節点の場合，節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とした時，$E_v$ のなかでただ一つの変数のみが 1 となる．
- 節点 $v$ が中間節点の場合，節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とした時，$E_v$ のなかで 1 となる変数は 0 個または 2 個である．
- 枝 $e$ に接続する両端の節点 $v_1$ と $v_2$ に対して，$e$ が 1 の時，$v_1 = v_2$ が成り立つ．

枝の変数は 2 値なのでそのまま命題論理式となるが，節点の変数は多値なので複数の 2 値変数を用いて符号化する必要がある．ここでは自然な 2 進符号化を用いるものとする．後述するように one-hot 符号化も考えられるが，この方式ではメリットがない．今，ラベルの最大値を $K$ とすると，$\log_2(K+1)$ 個の 2 値変数を用いてラベル値を表すことができる．ここでは $v_1$ の値を表す 2 値変数を $v_1^0, v_1^1, \ldots, v_1^{\log_2(K+1)}$ とする，$v_2$ に関しても同様とする．すると，上の制約は以下の命題論理式となる．

- 終端節点に接続する枝の制約

  $$\bigwedge_{e_i, e_j \in E_v} (\neg e_i \vee \neg e_j) \wedge \bigvee_{e_k \in E_v} (e_k)$$

  前半の 2 項節は同時に 2 つの枝が 1 とならないという制約であり，最後の節は最低 1 つの枝が 1 となる制約である．
- 中間節点に接続する枝の制約．一般的なケースを和積形論理式で書くと複雑になるので要素数が 2 から 4 までの場合の式を書く．

  $E_v = \{e_1, e_2\}$ の時

  $$(\neg e_1 \vee e_2) \wedge (e_1 \vee \neg e_2)$$

  $E_v = \{e_1, e_2, e_3\}$ の時

  $$\begin{aligned} &(\neg e_1 \vee \neg e_2 \vee \neg e_3) \\ \wedge\ &(\neg e_1 \vee e_2 \vee e_3) \\ \wedge\ &(e_1 \vee \neg e_2 \vee e_3) \\ \wedge\ &(e_1 \vee e_2 \vee \neg e_3) \end{aligned}$$

  $E_v = \{e_1, e_2, e_3, e_4\}$ の時

  $$\begin{aligned} &(\neg e_1 \vee e_2 \vee e_3 \vee e_4) \\ \wedge\ &(e_1 \vee \neg e_2 \vee e_3 \vee e_4) \\ \wedge\ &(e_1 \vee e_2 \vee \neg e_3 \vee e_4) \\ \wedge\ &(e_1 \vee e_2 \vee e_3 \vee \neg e_4) \\ \wedge\ &(e_1 \vee \neg e_2 \vee \neg e_3 \vee \neg e_4) \\ \wedge\ &(\neg e_1 \vee e_2 \vee \neg e_3 \vee \neg e_4) \\ \wedge\ &(\neg e_1 \vee \neg e_2 \vee e_3 \vee \neg e_4) \\ \wedge\ &(\neg e_1 \vee \neg e_2 \vee \neg e_3 \vee e_4) \end{aligned}$$

  和積形論理式のため直感的には理解しづらいが，各々のケースで 0 個または 2 個の変数が 1 となる組み合わせを具体的に列挙してそれを式の形で書き表している．
- 隣接する 2 つの変数に関する制約

  $$\bigwedge_{i=0}^{\log_2(K+1)} (\neg e \vee v_1^i \vee \neg v_2^i) \wedge (\neg e \vee \neg v_1^i \vee v_2^i)$$

[*1] [3] ではグラフの枝に向きを持たせているが，本稿の方式では無向グラフを用いている．

各々ビットごとに同じ形の式 $e \to (v_1^i = v_2^i)$ を和積論理式の形で表したものである．

なお，[3] と同様に，これだけの制約では実際には終端節点と連結していない冗長な線分 (ループ) が形成される恐れがある．これに関しては解を出力する際に取り除くこととする．

### 3.2 方式 2: 枝に多値変数を割り当てる方式

実は方式 1 の変数のうち，真の決定変数となっているのは枝の 2 値変数だけで，節点の多値変数は枝の 2 値変数の結果から従属的に定められる．しかし，枝の 2 値変数のみでは異なる線分が交わらない，という制約を表すことが難しい．そこで，枝の変数を多値変数とし，枝にラベル値を持たせる方式を考える．この場合，考慮する制約は以下のようになる．

- 節点 $v$ が終端節点の場合，節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とした時，$E_v$ のなかでただ一つの変数のみが $v$ のラベル値と等しくなり，残りは変数は 0 となる．
- 節点 $v$ が中間節点の場合，節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とした時，$E_v$ の変数が全て 0 か 2 つの変数が非 0 で等しい値となる．

枝の多値変数を 2 値に符号する方式としては方式 1 と同様の 2 進符号化も考えられるが，$K$ 個の変数を用いた one-hot 方式を用いることもできる．本稿ではこの 2 種類の方式をどちらも実装し，比較を行った．

実は 2 進符号化を用いる方式では，枝が線分の径路として選ばれているかどうかを表す 1 ビットの変数を設けている．以降，この変数を選択変数と呼ぶ．これはラベルの値が 0 かどうか ($e^0, e^1, \ldots, e^{\log_2(K+1)}$ が全て 0 かどうか) を表すものなので従属的な変数であるが論理式を簡単にするために導入している．この変数に関する制約は方式 1 の枝の変数の制約と同一となる．次はこの選択変数とラベル値を表す変数の制約である．

- 終端節点 $v$ に隣接する枝の制約
  節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とする．枝 $e_1$ に関する選択変数を $s_1$，ラベル値を表す変数を $l_1^0, l_1^1, \ldots, l_1^{\log_2(K+1)}$ とする．また，終端節点 $v$ のラベル値の 2 進表現を $b^{\log_2(K+1)} \ldots b^1 b^0$ とする．

$$\bigwedge_{e_i \in E_v} \bigwedge_{j=0}^{\log_2(K+1)} (\neg s_i \vee l_i^j \oplus \bar{b^j})$$

  まず，選択変数 $s_i$ が 0 ならば何の制約も必要ない．$s_i$ が 1 の時，つまり，$v$ に接続した枝 $e_i$ が選ばれている時にはそのラベル値が $v$ のラベル値と一致する必要がある．上記の式中で排他的論理和が現れるが，実際には $b^j$ は 0 か 1 かの定数なので $l_i^j$ の極性を表している．
- 中間節点 $v$ に隣接する枝の制約
  上記と同様に，節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とする．枝 $e_1$ に関する選択変数を $s_1$，ラベル値を表す変数を $l_1^0, l_1^1, \ldots, l_1^{\log_2(K+1)}$ とする．

$$\bigwedge_{e_{i1}, e_{i2} \in E_v} \bigwedge_{j=0}^{\log_2(K+1)} (\neg s_{i1} \vee \neg s_{i2} \vee l_{i1}^j \vee \neg l_{i2}^j) \wedge (\neg s_{i1} \vee \neg s_{i2} \vee \neg l_{i1}^j \vee l_{i2}^j)$$

  つまり，$e_{i1}$ と $e_{i2}$ が選ばれているには双方のラベル値が同じでなければならない．

one-hot 符号化の場合もほぼ同様であるが，個々のラベル値ごとの変数が選択変数と同様の意味を持つので選択変数は必要ない．そのため，制約式は簡単になっている．

- 終端節点 $v$ に隣接する枝の制約
  節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とする．枝 $e_1$ のラベルを表す変数を $l_1^1, l_1^1, \ldots, l_1^K$ とする．終端節点 $v$ のラベル値を $d$ とする．すると $l_1^d, l_2^d, \ldots$ のうち 1 つだけが 1 となるという式と $d$ 以外の変数 $l_i^j (j \neq d)$ が 0 であるという式を作ればよい．
- 中間節点 $v$ に隣接する枝の制約
  上記と同様に，節点 $v$ に接続する枝の集合を $E_v = \{e_1, e_2, \ldots\}$ とする．枝 $e_1$ のラベル値を表す変数を $l_1^1, l_1^2, \ldots, l_1^K$ とする．すると個々の $j = 1, 2, \ldots, K$ ごとに，$l_i^j$ が全て 0 であるか 2 つだけ 1 であるという式を作ればよい．

### 3.3 方式 3: 節点に多値変数を割り当てる方式

この方式は方式 2 とは逆に方式 1 から枝の 2 値変数を取り除いたものである．ナンバーリンクの解を線分の形ではなく，各マス目に書き込まれた数字だと思えば，この方式が最もその形に近いものとなる．この方式の制約は以下のようになる．

- 節点 $v$ が終端節点の場合，$v$ の値はラベル値となる．$v$ に隣接した節点のうち，$v$ と同じ値を持つ節点はただ 1 つである．残りの節点値はなんでもよい．
- 節点 $v$ が中間節点の場合，$v$ の値は 0 であるか，$v$ に隣接した節点のうち 2 つの節点と同じ値となる．

厳密にはこの制約を設けると図 2 のような径路を見逃すことになるが，線分長の短い解が望ましいのであればそもそもこのような径路を見つける必要がないので問題はない．

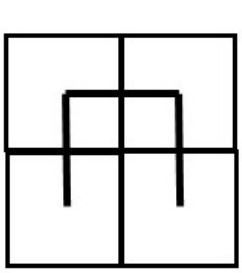

**図 2** 迂回径路

節点の多値変数を 2 値に符号化する方式としては方式 2

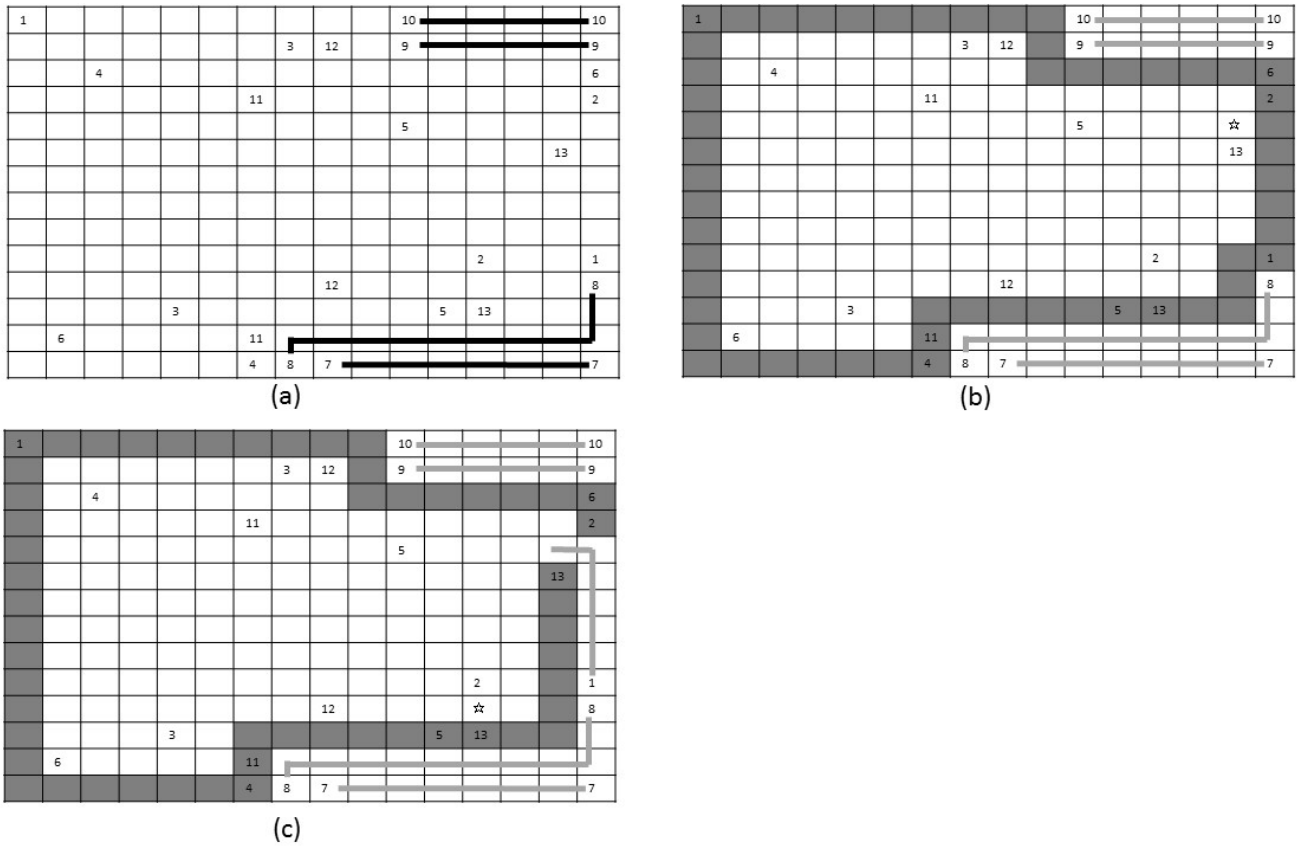


**図 3** 外周ヒューリスティック

と同様に 2 進符号化と one-hot 符号化が考えられるが，この方式の場合 2 進符号化は，中間節点に隣接する 2 つの節点とラベル値が等しい，という制約を表すのが複雑になると考えて採用しなかった．

- 終端節点 $v$ に隣接する枝の制約
  節点 $v$ に隣接する節点の集合を $V_v = \{v_1, v_2, \ldots\}$ とする．節点 $v_1$ のラベルを表す変数を $l_1^1, l_1^2, \ldots, l_1^K$ とする．終端節点 $v$ のラベル値を $d$ とする．すると $l_1^d, l_2^d, \ldots$ のうち 1 つだけが 1 となるという式を作ればよい．
- 中間節点 $v$ に隣接する枝の制約
  上記と同様に，節点 $v$ に隣接する節点の集合を $V_v = \{v_1, v_2, \ldots\}$ とする．節点 $v_1$ のラベルを表す変数を $l_1^1, l_1^2, \ldots, l_1^K$ とする．節点 $v$ のラベルを表す変数を $l^1, l^2, \ldots, l^K$ とする．すると個々の $j = 1, 2, \ldots, K$ ごとに，$l^j$ が 1 であれば $l_i^j$ のうちで 2 つだけ 1 であるという式を作ればよい．

## 4. 外周ヒューリスティック

外周部に終端節点がある場合，探索を行わずに径路が確定する場合がある．ここではそのようなケースに着目したヒューリスティックについて提案する．

図 3 にナンバーリンク問題の例を示す [*2]．この例では外周部に 1, 10, 10, 9, 6, 2, 1, 8, 7, 7, 8, 4 という終端節点が並んでいる (左上隅から時計回り)．このなかで，10 と 7 というラベルを持つ節点は間に他の終端節点を挟まずにならんでいる．そこで，このように隣接した同番号の終端節点を外周にそって確定させる．すでに確定したマス目を枠外とみなして新しい外周を考えると同様に 9 と 7 が隣接していることがわかる．そこで，9 と 7 も新しい外周に沿って結ぶ (図 3(a))．ここではこのように外周上で連続している終端節点を結ぶ処理を「ステップ 1」と呼ぶことにする．

図 3(b) はステップ 1 を繰り返し適用し変化のなくなった状態である．新しい外周は黒く塗りつぶされたマス目である．外周上の終端節点は 1, 6, 2, 1, 13, 5, 11, 4 であり，連続した終端節点はない．ただ，1 のラベルを持つ終端節点が 2 つとも外周上に現れている．ということは，1 と 1 を結んだ時に 6 と 2 の線分は上側に，13,5,11,4 の線分は下側になければならないことになる．一方，13 の外周上でない節点は右上方 (15,6) にある．13 の線分は 1 の線分よりも下になければならないので，この時点で 1 の線分は必ず 13 の上のマス目 (15,5) を通ることになる．図 3(b) 中の☆印がそのマス目である．すると，右下の 1 から☆印までの径路が確定する．この処理を「ステップ 2」と呼ぶ．

図 3(c) にステップ 2 までが終了した状態を示す．ここで右上の 2 のマス目 (16,4) に着目する．2 の線分は必ずその左隣のマス目を通らなければならない．このように終端

[*2] DA シンポジウム 2014 アルゴリズムデザインコンテストの NL_Q07

に隣接するマス目で利用可能なものが唯一になった場合には，そのマス目に線分を伸ばす．この処理を「ステップ 3」と呼ぶ．この場合，2 の直上の 6 も同様である．その結果，新たな外周上に 13 の節点が隣接することになる (ステップ 1)．また，2 の線分が 1 の線分より上になければならない，という制約より 1 の線分が図 3(c) の☆印を通らなければならないことがわかる (ステップ 2)．また，ステップ 3 により右上の 6 と 2 の配線が少しずつ確定してゆく．

この例ではこのステップ 1，ステップ 2，ステップ 3 を交互に繰り返すことで全ての径路が確定する．後述するが，この問題はエンコーディング方式によってはかなりの計算時間を必要とするのでこのヒューリスティックは効果的であると言える．ただし，実はステップ 1 はただのヒューリスティックであり，正解を見逃す可能性がある．例えば図 4 の例で，外周上の節点は 1, 2, 2, 1 であり，提案手法のステップ 1 によって線分 1 は図中の A の様に結ばれる．ところが，同図 B のような経路を用いることも可能である．A の経路を用いたために他の線分に対する結線が不可能な場合もあるため，このヒューリスティックを用いた結果，解けなかった場合には，このヒューリスティックを適用せずに問題を解き直す必要がある．

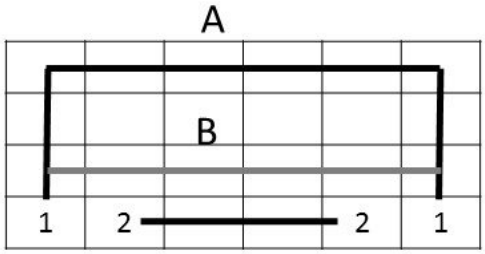


**図 4** 外周ヒューリスティックの反例

## 5. 実験結果

以上の方式を実装し，実験を行った．例題は DA シンポジウム 2014 アルゴリズムデザインコンテストのオリジナル問題を用いた [*3]．前半の小さな問題は省略している．使用計算機は Intel Core i7-2600 (3.40GHz)(メモリ 16GB)，OS は FreeBSD 10.2-RELEASE，使用コンパイラは clang++-3.4 である．プログラムはシングルスレッドであり，複数のコアは用いていない．また，SAT ソルバは glueminisat2[*4] を用いている．表 1 に結果を示す．ここで，方式 2A は方式 2 の 2 進符号化，方式 2B は方式 2 の one-hot 符号化を表す．外周に基づくヒューリスティックは用いていない．

方式 3 が圧倒的に良い結果を得ている．また，同じ方式 2 でも one-hot 符号化のほうが 2 進符号化よりも速い場合が多い．これは one-hot 符号化に関する制約節の多くが 2 項節 (2 つのリテラルで構成される和項) であることに起因するものと思われる．同様の傾向は [1], [2] でも報告されている．SAT ソルバを用いる際のノウハウとして有益であると思わる．また，論理式の変数の数，節の数，リテラル数を比較すると方式 1 が最も少ないにもかかわらず，処理時間は方式 3 が最も短くなっている．単純に変数やリテラル数が少ない論理式が早く解けるわけではないことを示している．

NL_Q07 は外周ヒューリスティックを用いると探索なしで経路が確定する．外周ヒューリスティックの処理時間は無視できるほど高速なので場合によっては効果的なヒューリスティックである．

[*3] http://www.sig-sldm.org/DC2014/ADC2014_QA.zip
[*4] http://glueminisat.nabelab.org/

## 6. おわりに

本稿では SAT ソルバを用いてナンバーリンク問題を解く場合のエンコーディング方式に関して比較検討を行った．結論としては本稿で方式 3 と呼ぶ方式—マス目の線分番号を表す多値変数を割り当てる方式—が最も効率的であることがわかった．また，外周に沿って経路を割り当てるヒューリスティックの提案を行った．問題によっては探索なしで解が求められるものもあり，効果的なヒューリスティックであると思われる．

ナンバーリンク問題の難しさは異なる線分の経路が交わらない，という制約をどのように命題論理式にエンコーディングするか，というところにあると考えられる．今回提案した以外のエンコーディング方式や述語論理式に定式化を行って SMT ソルバを用いて解くなどのアプローチも今後の検討課題である．

### 参考文献

[1] 松永裕介：LUT 回路のブーリアンマッチング手法について，電子情報通信学会技術報告 vol. 113, no. 416, VLD2013-127, pp. 149–154 (2014).
[2] Matsunaga, Y.: Accelerating SAT-based Boolean Matching for Heterogeneous FPGAs using One-hot encoding and CEGAR technique, *Proceedings of of Asia and South Pacific Design Automation Conference 2015*, pp. 255–260 (online), DOI: 10.1109/ASPDAC.2015.7059014 (2015).
[3] 田村直之，宋　剛秀，番原睦則，鍋島英知：SAT 型制約ソルバーを用いたナンバーリンクの解法，DA シンポジウム 2014，情報処理学会，pp. 215–220 (2014).

**表 1** 各種エンコーディング方式の比較結果

| | | 方式 1 | 方式 2A | 方式 2B | 方式 3 |
|---|---|---|---|---|---|
| NL_06 | CPU 時間 | 3 分 59.93 秒 | NA(1 時間以上) | 30 分 23.79 秒 | 1.04 秒 |
| | 変数の数 | 1480 | 2360 | 4248 | 2268 |
| | 節の数 | 5512 | 11922 | 109949 | 22462 |
| | リテラル数 | 17597 | 46149 | 244882 | 70676 |
| NL_07 | CPU 時間 | NA(1 時間以上) | NA(1 時間以上) | 8 分 24.63 秒 | 1.94 秒 |
| | 変数の数 | 1498 | 2390 | 6214 | 3315 |
| | 節の数 | 5541 | 12103 | 216977 | 37605 |
| | リテラル数 | 17521 | 46493 | 467568 | 110908 |
| NL_08 | CPU 時間 | 0.53 秒 | 0.68 秒 | 0.42 秒 | 0.04 秒 |
| | 変数の数 | 2093 | 3438 | 16044 | 8512 |
| | 節の数 | 7257 | 16172 | 843309 | 131968 |
| | リテラル数 | 21443 | 59212 | 1727563 | 321726 |
| NL_09 | CPU 時間 | 5.71 秒 | NA(1 時間以上) | 15.47 秒 | 0.03 秒 |
| | 変数の数 | 2232 | 3672 | 11016 | 5832 |
| | 節の数 | 8136 | 18298 | 497563 | 77399 |
| | リテラル数 | 25141 | 69608 | 1048499 | 213947 |
| NL_13 | CPU 時間 | 20 分 22.21 秒 | NA(1 時間以上) | 2 分 56.42 秒 | 0.04 秒 |
| | 変数の数 | 2607 | 4302 | 12189 | 6426 |
| | 節の数 | 9638 | 21763 | 539051 | 84577 |
| | リテラル数 | 30064 | 83342 | 1140247 | 237085 |
| NL_14 | CPU 時間 | 24 分 40.79 秒 | 4 分 45.74 秒 | 1 分 34.42 秒 | 59.50 秒 |
| | 変数の数 | 2607 | 4302 | 12189 | 6426 |
| | 節の数 | 9638 | 21763 | 539051 | 84233 |
| | リテラル数 | 30064 | 83342 | 1140247 | 231237 |
| NL_15 | CPU 時間 | NA(1 時間以上) | NA(1 時間以上) | 3 分 36.63 秒 | 0.23 秒 |
| | 変数の数 | 5266 | 8932 | 44660 | 23275 |
| | 節の数 | 19408 | 45395 | 3889802 | 48889 |
| | リテラル数 | 59586 | 172983 | 7994901 | 1220929 |