A-003

二分決定グラフを用いた二次疑ブール関数の最小解列挙
Enumerating Minimum Solutions of Quadratic Pseudo Boolean functions with decision diagrams

松尾 毬花[1)]　**川原 純**[1)]　**湊 真一**[1)]
Marika Matsuo　Jun Kawahara　Shin-ichi Minato

1 はじめに

ZDD (Zero-suppressed Binary Decision Diagram; ゼロサプレス型二分決定グラフ) [6] は組合せ集合をコンパクトに効率的に表現するために用いられる，有向非巡回グラフによって表されるデータ構造である．本研究では，ZDD によって二次疑ブール関数の定義域が与えられる場合に，二次疑ブール関数の最小解を求め，二次関数の重みが最小となる集合をすべて含む ZDD を構築する手法を提案する．二次疑ブール関数とは，二つのブール変数 x_i，x_j に対して重み w_{ij} が与えられたとき，$\sum_{i,j} w_{ij}x_ix_j$ の形で表される関数である．二次疑ブール関数の最小化は，最大カット問題や最大クリーク問題など多くの NP 困難問題を含み，施設割当問題などの二次割当問題 [4] やタンパク質構造推定 [5]，木材 CAD における木材の配置など様々な応用がある．問題の変数の制約が ZDD で表現されていれば，提案手法により二次疑ブール関数の最小化が可能となり，さらに最小な解の列挙も可能となる．提案アルゴリズムの性能を計算機実験により評価した．

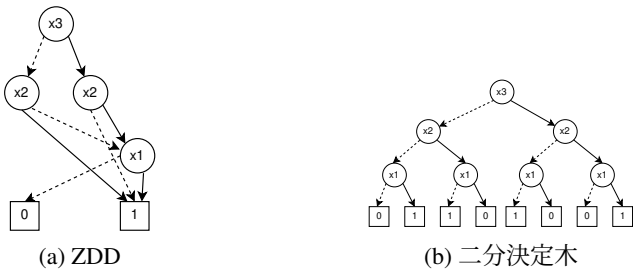
2 準備

2.1 ZDD

ZDD [6] は，組合せ集合の表現に用いられる，連結な有向非巡回グラフである．ZDD には，終端節点と非終端節点の 2 種類の節点がある．終端節点は出次数が 0 の節点である．終端節点は 2 つ存在し，それぞれ 0 終端節点，1 終端節点と呼ばれる．終端節点は出次数が 0 である．0 終端節点は空集合を表し，1 終端節点は空集合 1 つからなる集合族を表す．非終端節点はそれぞれ集合の要素 $x_1,\ldots,x_n$ のいずれかをラベルとして持ち，2 つの子節点への有向枝を持つ．この有向枝は 1 枝，0 枝と呼ばれ，それぞれラベルの要素を選ぶ，あるいは選ばないことを表す．根から順に枝をたどり，0 終端節点に到達したとき，対応する集合は ZDD が表す集合族には含まれないことを意味する．1 終端節点に到達したとき，対応する集合は ZDD が表す集合族に含まれることを意味する．またラベルが x_i の節点から出る枝がラベルが x_j の節点を指すとき，$i > j$ が成り立つ．

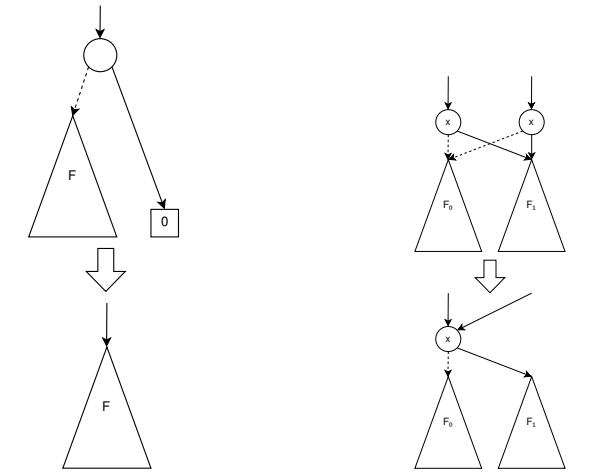
ZDD の例を図 1a に示す．この ZDD は集合 $\{\{x_1\},\{x_2\},\{x_3\},\{x_1,x_2,x_3\}\}$ を表す．丸い枠は非終端節点を表し，枠内の数字が集合の要素を表す．四角の枠は終端節点を表し，枠内の数字が 1 のものは 1 終端節点，0 のものは 0 終端節点を表す．実線の矢印が 1 枝を，破線の矢印が 0 枝を表す．

ZDD は二分決定木を以下に述べる圧縮規則により圧縮することで得られる．例えば図 1a の圧縮前の二分決定木は図 1b のようになる．二分決定木の圧縮規則には，冗長節点の削除と等価節点の共有の 2 つがある．冗長節点とは 1 枝が 0 終端節点を指している節点のことである．冗長節点を削除する場合，この節点を指していた枝は削除された節点の 0 枝の終点に接続する．冗長節点の削除の例を図 2a に示す．等価節点とは 1 枝の終点が同じかつ 0 枝の終点が同じかつ 2 つの節点のラベルが同じ，2 つの非終端節点のことである．等価節点を共有する場合，片方の節点を削除し，この節点を指していた枝はもう片方の節点に接続する．等価節点の共有の例を図 2b に示す．これらの規則をできる限り適用することで ZDD が得られる．



(a) ZDD　(b) 二分決定木

図 1: ZDD と二分決定木の例



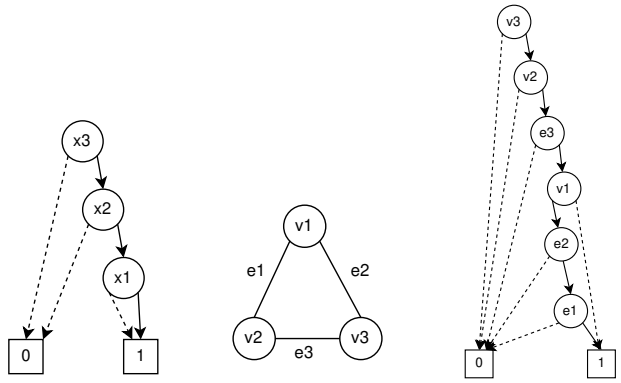
(a) 冗長節点の削除　(b) 等価節点の共有

図 2: ZDD の圧縮規則

2.2 ZDD による線形重み最小化

ZDD を用いるとき，ZDD の各変数に重みを設定すると，ZDD が表現する集合の中から，重みの和が最小・最大となる集合を取得することができる [1, 6]．n 変数 $x_1,x_2,\ldots,x_n$ の集合族を表す ZDD z と重み $w_1,w_2,\ldots,w_n$ が与えられるときを考える．このとき，$\{x_1,x_2,\ldots,x_n\}$ の部分集合を S とするとき，S の集合の重みを $\sum_{i=1}^{n} w_ix_i$ で定義する．ここで，x_i が S に含まれるときは $x_i = 1$ であり，含まれないときは $x_i = 0$ であるとする．z に含まれる集合のうち，集合の重みの最小値（または最大値）とその重みをもつ集合（の 1 つ）を取得することができる [1, 6]．区間メモ化探索技法 [7] を用いると，重みの和が指定した値以上・以下・より大きい・より小さい集合

1) 京都大学大学院情報学研究科



(a) ZDD P の例　(b) グラフ G　(c) ZDD Q

図 3: ZDD P とグラフ G と ZDD Q

の集合族となるような ZDD を取得することができる．

3 提案手法

n 変数 $x_1, x_2, \ldots, x_n$ の集合族を表す ZDD z と重み $w_{12}, w_{13}, \ldots, w_{1n}, w_{23}, w_{24}, \ldots w_{(n-1)n}$ が与えられるときを考える．このとき，$\{x_1, x_2, \ldots, x_n\}$ の部分集合を S とするとき，S の集合の重みを $\sum_{i,j} w_{ij} x_i x_j$ で定義する．ここで，x_i が S に含まれるときは $x_i = 1$ であり，含まれないときは $x_i = 0$ であるとする．このとき，ZDD に含まれる集合のうち，集合の重みの最小値（または最大値）とその重みをもつ集合（の 1 つ）を取得する手法を提案する．この計算を二次疑ブール関数の重み最適化と呼ぶ．

ブール変数を頂点とするようなグラフ G を以下の通り構築する．G の頂点 v_i はブール変数 x_i に対応する．2 つのブール変数 x_i，x_j の重み w_{ij} が 0 でないとき，G では対応する 2 つの頂点を辺で結ぶ．すなわち，G の頂点集合は $\{v_1, v_2, \ldots, v_n\}$ であり，辺集合は $E = \{(v_i, v_j) \mid 1 \le i < j \le n, w_{ij} \neq 0\}$ である．

ZDD P が入力として与えられるとする．P が表現する集合族に含まれる集合のうち，重みが最小になるものを列挙する手法を示す．まず P の各ラベルが G の頂点に対応すると考える．そして P の各集合 S によって誘導される G の誘導部分グラフ $G'[S]$ を考える．各 S について，S と $G'[S]$ の和集合を集めてできる集合族を ZDD として表現し，それを Q とする．この Q に対して，$x_1, \ldots, x_n$ の重みを 0，E の各辺の重みを w_{ij} として線形重み最小化を行うことで，二次疑ブール関数の最小化ができる．

例えば，図 3a のように $\{\{x_1, x_2, x_3\}, \{x_2, x_3\}\}$ を表す ZDDP と重み $w_{12} = 1, w_{13} = 2, w_{23} = 3$ が与えられたとする．このとき変数 x_1, x_2, x_3 をそれぞれ頂点 v_1, v_2, v_3 とみなすグラフ G は図 3b のようになる．P と G に対して頂点と辺の集合を表す ZDD Q を構築した結果を図 3c に示す．Q は集合族 $\{\{v_1, v_2, v_3, e_1, e_2, e_3\}, \{v_2, v_3, e_3\}\}$ を表す．さらに頂点 v_1, v_2, v_3 に対して重みを 0，辺 e_1, e_2, e_3 に対してそれぞれ重み $1, 2, 3$ を設定し，線形重み最小化を行うことで二次疑ブール関数の最小の重み 3 と重みが最小となる集合 $\{v_2, v_3\}$ を求めることができる．

ZDD Q から，重み最小となる集合のみを抽出するには，区間メモ化探索技法 [7] を用いる．最小値を閾値とすることで，重みが最小となる集合をすべて抽出できる．

ZDD では，根から終端節点までの経路において変数ラベルが出現する順番は固定しなければならない．ZDD Q では頂点だけを見たときの変数順は $v_n, v_{n-1}, \ldots, v_1$ となり，その間に辺の変数が入る．辺変数 $e_i = (v_\alpha, v_\beta)$ $(\alpha > \beta)$ は頂点変数 v_β と $v_{\beta-1}$ の間に現れる．さらに，v_β と $v_{\beta-1}$ の間に辺 $e_{i1}, e_{i2}, \ldots, e_{it}$ がこの順で現れるとする．ここで，$e_{ij} = (v_{\alpha(j)}, v_\beta)$ である．このとき，$\alpha(1) > \alpha(2) > \cdots > \alpha(t)$ が成り立たなければならない．

最初に ZDD P を考慮せず，すべての誘導部分グラフの集合を表す ZDD の構築法について述べる．ZDD はトップダウン構築法と呼ばれる方法により，根から終端に向かう方向に順に節点を作成する．最初に根節点を作成し，根節点の 0 枝側の節点と 1 枝側の節点を作成する．以降は幅優先的に各節点の 0 枝側と 1 枝側の節点を作成する．このように節点を作成する際，ある節点 v に着目すると，根節点から v までの経路は 1 つの部分集合（経路上で 1 枝を選択した節点のラベルの集合）に対応する．v のラベルが v_i で，G は辺 $e = (v_j, v_i)$ $(j > i)$ を持つとし，ラベルが v_j の節点で 1 枝を選択したとすると，v の 1 枝は v_i を選択することを意味し，v_j と v_i の両方を使用することを意味する．このとき，辺 e は必ず使用しなければならない．v の 1 枝の先の節点を作成し（v_1 とする），v_1 のラベルを e とし，v_1 の 0 枝の先は 0 終端に接続する．v_β と $v_{\beta-1}$ の間に辺 $e_{i1}, e_{i2}, \ldots, e_{it}$ が現れるとすると，任意の $j = 1, \ldots, t$ について $e_{ij} = (v_{\alpha(j)}, v_\beta)$ という辺が存在するので，同様にラベルが e_{ij} である節点を作成し，その 0 枝の先を 0 終端に接続する．それらの節点を 1 枝で接続する．

辺 $e = (v_j, v_i)$ $(j > i)$ について，v_j と v_i が使用される場合，またその場合に限り e を使用する．したがって，v_j を選択した（ラベルが v_j の節点から 1 枝の先を作成する）ときに，後に v_i の使用の有無により e を使用するべきかどうかの選択が変わる．このため，各節点には辺の候補集合 $edges$ を記憶させ，v_j を使用した時点で v_j に接続するすべての辺を $edges$ に追加する．

次に ZDD P を考慮して，P に含まれる集合だけを Q に含めるための方法について述べる．これはサブセッティング法 [2] と呼ばれる考え方を用いる．Q のトップダウン構築時に，Q の各節点には，$edges$ に加えて P の節点 var を格納する．Q の根節点の var には P の根節点を格納する．以降，Q の節点 v の i 枝（$i = 0, 1$）の先の節点 v_i を作成する場合，v_i の var は，v の var の i 枝の先であるとする．最終的に var は 0 終端か 1 終端となるが，1 終端になる（すなわち P の集合となっている）場合に限り Q でも 1 終端とする．ZDD P を ZDD Q へ変換するアルゴリズムを Algorithm 1 に示す．

このアルゴリズムを適用すると，P に含まれない解は，Q にも含まれない．まず，P に含まれる集合に対して，その集合に含まれない節点を選択すると，その節点のラベルと節点番号が一致しないため，1 枝は 0 終端に接続される．また，P に含まれない集合に対しては，その集合を Q の根から辿っていくと節点は必ず 0 終端になるため，0 終端に接続される．また P に含まれる解は，その集合に含まれるラベルをもつ節点の節点が 0 終端になることはないので，枝刈りが行われることはなく，Q にも含まれる．

Algorithm 1 ZDD P から Q を構築するアルゴリズム

Q の根節点 n を作成
$n.lab \leftarrow v_l$, $n.var \leftarrow n_p$, $n.edges \leftarrow \emptyset, U_{v_l} \leftarrow \{n\}$
/* $n.lab$ は節点 n のラベル、n_p は P の根節点 */
$U_{v_i} \leftarrow \emptyset$ for $i = 1, \ldots, l-1$
for $i = l, l-1, \ldots, 1$ **do**
  /* U_{v_i} はラベルが v_i の節点の集合 */
  for $n \in U_{v_i}$ **do**
    $\beta \leftarrow n.var$
    for $x \in \{0, 1\}$ **do**
      節点 n の x 枝側の節点 n_x を作成
      $\beta_x \leftarrow$ GetNode(β, x) /* 節点 β の x 枝先 */
      /* $\beta_x.lab$ は節点 β_i のラベル */
      if $if v_{i-1} = \beta_x.lab$ **then**
        $n_x.var \leftarrow \beta_x$
      else
        if $x = 1$ **then**
          0 終端に接続
      if $n_x.var = 0$ **then**
        枝刈りを行い，0 終端に接続
      $E' \leftarrow$ GetEdges(v_i) /* v_i に接続する辺 */
      if $x = 0$ **then**
        $n_0.edges \leftarrow n.edges \setminus E'$
        if $n'_x = n_x$ を満たす n'_x が存在 **then**
          節点 n_x を削除
          x 枝を n'_x に接続
        $U_{v_{i-1}} \leftarrow U_{v_{i-1}} \cup \{n_x\}$
        if $i = 1$ **then**
          if $n_0.edges = \emptyset$ & $n_0.var = 1$ **then**
            1 終端に接続
          else
            0 終端に接続
      else if $x = 1$ **then**
        $\{e_{i1}, \ldots e_{it}\} = \{e \in n.edges \mid v_i$に接続 $\}$
        節点 $n_{1u} (u = 1, 2, \ldots, t+1)$ を作成
        $n_{1u}.var \leftarrow n_x.var$
        for $u = 1, 2, \ldots, t$ **do**
          $n_{1u}.lab \leftarrow e_{iu}$
          1 枝は $n_{1(u+1)}$ に接続
          0 枝は 0 終端に接続
          $n_{1u}.edges \leftarrow n_{1(u-1)}.edges \setminus \{e_{iu}\}$
        if $i > 1$ **then**
          $n_{1(t+1)}.lab \leftarrow v_{i-1}$
          $n_{1(t+1)}.edges \leftarrow n_{1t}.edges$
            $\cup \{e \in E' \mid e \notin n.edges\}$
          if $n_x = n_{1(t+1)}$ を満たす n_x が存在 **then**
            節点 $n_{1(t+1)}$ を削除
            節点 n_t の 1 枝を $n'_{1(t+1)}$ に接続
          $U_{v_{i-1}} \leftarrow U_{v_{i-1}} \cup \{n_{1(t+1)}\}$
        else
          if $n_{1t}.edges = \emptyset$ **then**
            1 終端に接続
          else
            0 終端に接続

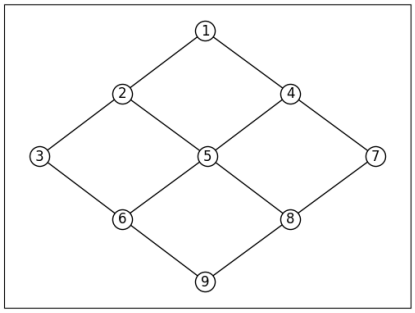


図 4: グリッドグラフの例

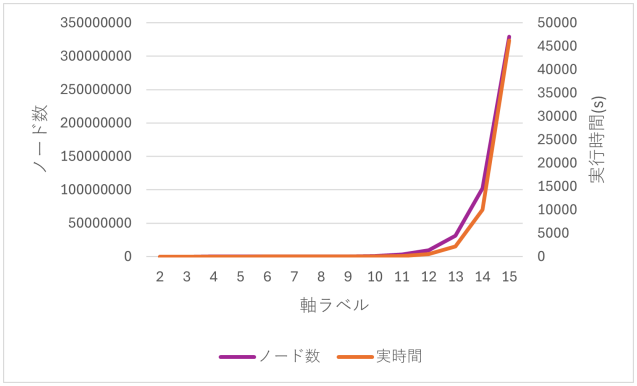


図 5: $n \times n$ のグリッドグラフに対する頂点と辺の集合を表す ZDD の節点数および実時間

4 実験結果

考案した手法について性能評価を行うため，計算機実験を行った．計算機は Linux PC (Intel Xeon Gold 6134，8 コア 3.20GHz，主記憶 3TB) を使用した．手法の実装は C++言語で行い，ZDD の構築には TdZdd [2] ライブラリを用いた．

4.1 頂点と辺の集合の ZDD の構築

実験データとして，以下を作成した．二次疑ブール関数の定義域はグリッドグラフの始点から終点までのパスの集合を表す ZDD である．パスは辺の集合で表され，ZDD ではパスの集合を表す．ZDD はフロンティア法 [3] によって構築した．グリッドグラフ上でパスが何回曲がるかを数える二次疑ブール関数を例として考える．二次疑ブール関数の重みは，グリッドグラフの曲がり角を構成する 2 つの辺に対して重みを 1，その他の辺に対して重みを 0 とした．例えば図 4 の 3×3 のグリッドグラフにおいて，始点は頂点 1，終点は頂点 9 とし，頂点 1 から頂点 9 までのパスを考える．また，重みについては辺 (1, 2) と辺 (2, 3) の 2 つが曲がり角を構成するため，この 2 つの辺を両方選択したとき重みが 1 となる．一方，辺 (1, 2) と辺 (2, 3) や辺 (1, 2) と辺 (3, 6) の 2 つを選択したときは曲がり角を構成しないため，重みは 0 となる．このときグリッドグラフの辺がグラフ G の頂点に，曲がり角を構成する 2 つの辺を結ぶ線が G の辺にあたる．このような G に対して頂点と辺の集合を表す ZDD を構築し，節点数と構築にかかる実時間について調べた．

まず，$n \times n$ のグリッドグラフに対して，頂点と辺の集合を表す ZDD を構築した結果を図 5 に示す．次に，$10 \times n$ のグリッドグラフに対して，頂点と辺の集合を表す ZDD を構築した結果を図 6 に示す．なお，どちら

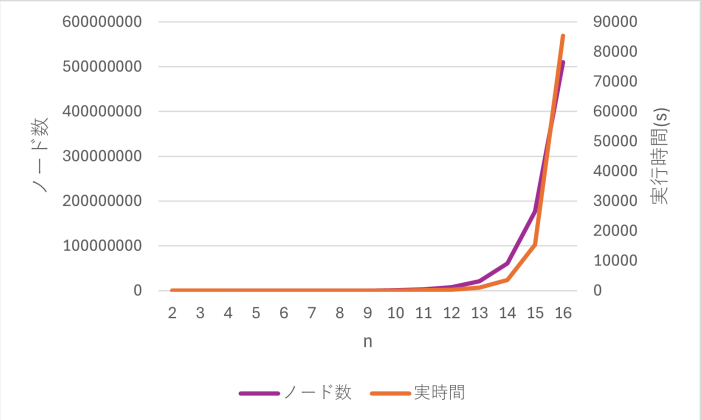


図 6: $10 \times n$ のグリッドグラフに対する頂点と辺の集合を表す ZDD の節点数および実時間

表 1: 10×10 のグリッドグラフに対する二次疑ブール関数の最小解列挙の結果

頂点と辺の集合の ZDD の構築時間	115 s
最小の重み	1
最小の重みを求める時間	2.65 s
最小の重みの ZDD の構築時間	1.60 s
最小の重みを持つ集合の数	2
辺を削除する時間	0.001 s

もこれ以上のサイズでは構築に 1 日以上かかりそうなため，実験は行わなかった．また，メモリの使用量は 15×15 のグリッドグラフに対しては約 29 GB，10×16 のグリッドグラフに対しては約 49 GB であった．

4.2 二次疑ブール関数の最小解列挙

10×10 のグリッドグラフに対して，曲がり角を構成する 2 つの辺に対して重みを 1，その他の辺に対して重みを 0 とした二次疑ブール関数の最小解を列挙した．この結果を表に示す．

この解は，グリッドグラフの始点から終点までのパスの中から曲がり角の個数を最小とするものを選ぶため，曲がり角の数が 1 となるパスを選ぶことになる．実際に，最小の重みは 1 となり，最小の重みを持つ集合は 2 つ存在することが確認できる．

4.3 パスの重みが最小となる集合の列挙

10×10 のグリッドグラフに対して，曲がり角を構成する 2 つの辺に対して重みを 5，その他の辺の重みを 0 から 10 までランダムに与えたときのパスの重みが最小となる集合を列挙した．この結果を表 2 に示す．

この例では辺の重みをランダムに与えているため，パスの重みが最小となる集合は 1 つとなったが，適当に与えれば複数見つけることも可能である．また最小解ではなく，パスの重みが指定した値以下となるような集合を列挙することも可能であり，この例で重みが最小+10 以下，つまり 99 以下となるような集合を列挙した場合，集合は 2 つ見つかった．

4.4 考察

4.2，4.3 節の結果から，提案手法は二次疑ブール関数の最小解を列挙することができることが確認できた．また，計算時間としては頂点と辺の集合の ZDD の構築に一番時間がかかり，ここがボトルネックになると考えら

表 2: 10×10 のグリッドグラフに対するパスの重みの最小解列挙の結果

頂点と辺の集合の ZDD の構築時間	115 s
最小の重み	89
最小の重みを求める時間	2.72 s
最小の重みの ZDD の構築時間	1.63 s
最小の重みを持つ集合の数	1
辺を削除する時間	0.001 s

れる．実際に，4.1 節の結果から，$n \times n$ のグリッドグラフでは n が 14，15 から，$10 \times n$ のグリッドグラフでは n が 15，16 から急激に時間がかかっている．図 5，6 からもわかるように，節点数と実行時間には相関が見られ，節点数が増えると実行時間も増加する．そのためパスの数が増えたり，重みをつける頂点のペアを増やしたりすると，構築により時間がかかる可能性がある．

4.5 関連研究

鈴木ら [8] は，与えられたグラフの部分グラフの集合が辺の集合族の ZDD として表現されている場合に，各辺集合に対して，辺に接続する頂点をすべて加えた ZDD を構築する方法を提案している．本稿で提案している手法は，頂点の集合族の ZDD に対して，（頂点が誘導する）辺を加えた ZDD を構築していると考えることができる．

5 おわりに

本研究では，ZDD を用いて二次疑ブール関数の最小解を列挙する手法を提案した．今後の課題として，二次疑ブール関数の最小化を行う他の手法との比較やより複雑なグラフに対しての評価が考えられる．

謝辞

本研究の一部は，JSPS 科研費 JP24H00690, JP25H01114 の助成を受けたものです．

参考文献

[1] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *Computers, IEEE Transactions on*, Vol. 100, No. 8, pp. 677–691, 1986.

[2] Hiroaki Iwashita and Shin-ichi Minato. Efficient top-down ZDD construction techniques using recursive specifications. *Hokkaido University Division of Computer Science TCS Technical Report TCS-TR-A-13-69*, 2013.

[3] Jun Kawahara, Takeru Inoue, Hiroaki Iwashita, and Shin-ichi Minato. Frontier-based search for enumerating all constrained subgraphs with compressed representation. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E100-A, No. 9, pp. 1773–1784, 9 2017.

[4] Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica*, Vol. 25, No. 1, pp. 53–76, 1957.

[5] Mark W. Lewis, Amit Verma, and Rick Hennig. Predicting 3D RNA folding patterns via quadratic binary optimization. 2021.

[6] Shin-ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proceedings of the 30th International Design Automation Conference*, pp. 272–277, 1993.

[7] Shin-ichi Minato, Jun Kawahara, Mutsunori Banbara, Takashi Horiyama, Ichigaku Takigawa, and Yutaro Yamaguchi. Fast enumeration of all cost-bounded solutions for combinatorial problems using ZDDs. *Discrete Applied Mathematics*, Vol. 360, pp. 467–486, 2025.

[8] 鈴木浩史, 湊真一. ZDD を用いたグラフ列挙索引化における頂点インデックスの追加. 人工知能学会研究会資料 人工知能基本問題研究会, Vol. 101, p. 08, 2016.