Vol.2022-AL-187 No.2
2022/3/14

ブロックDAGに対する最大k-独立集合問題の二分決定グラフを用いた解法

伝住 周平[1,a)]　川原 純[2,b)]

概要：DAG 型ブロックチェーンにおいて，誠実なブロックを特定する高速なアルゴリズムの設計は重要である．誠実なブロックを特定する問題は，大きな k-独立集合を求める問題として定式化されるが，k-独立集合問題は NP 困難であり，高速なアルゴリズム設計が難しい．本研究では，決定グラフを用いて大きな k-独立集合を複数求める高速な手法を提案する．計算機実験によりアルゴリズムの性能を検討する．

Solving the k-independent set problem for BlockDAG using decision diagrams

1. はじめに

0 以上の整数 k について，グラフに対する k-独立集合とは，グラフの頂点部分集合 S であり，S による誘導部分グラフの次数が高々 k となる．$k = 0$ のときはよく知られる独立集合の定義と一致する．本稿では，0 以上の整数 k と頂点に重みの付いたグラフが入力として与えられた際に，重みが最大となる k-独立集合を求める問題を考える．この問題は NP 困難である．

本問題は DAG（非巡回有向グラフ，Directed Acyclic Graph）型ブロックチェーンのブロック信頼性判定の際に現れる．ブロックチェーンとは改ざんが困難な分散型台帳である．DAG 型ブロックチェーンでは，データをブロックと呼ばれる構造に格納する．各々の参加者はブロックを生成し，既存のブロックを複数個選択し，生成したブロックから選択したブロックへ（分散的に）有向枝を張る．その結果，ブロックを頂点みなし，ブロックと有向枝からなる有向グラフは DAG となる．この DAG において，DAG を定められた変換ルールによって無向グラフに変換し，その無向グラフ上で k-独立集合問題を解くことで，悪意のない参加者が作成した可能性が高い，「誠実な」ブロックと呼ばれるブロックの検出を行うプロトコルが提案されている [8].

本研究では，決定グラフを用いて大きな k-独立集合を複数求める高速な手法を提案する．計算機実験によりアルゴリズムの性能を検討する．

2. 準備

2.1 k-独立集合問題

頂点集合 V，辺集合 E からなる単純無向グラフ $G = (V, E)$ を考える．本稿では入力グラフに対する記号を $G = (V, E)$ に固定する．入力グラフは連結であると仮定する（非連結である場合は連結成分ごとに考えればよい）．頂点集合 $S \subseteq V$ について，S を頂点集合として，以下の条件を満たす部分グラフを誘導部分グラフと呼び，$G[S]$ と表記する．

任意の頂点 $u, v \in S$ について $\{u, v\} \in E$ ならば，
またそのときに限り $G[S]$ は辺 $\{u, v\}$ を持つ．

0 以上の整数 k と，頂点集合 $S \subseteq V$ について，$G[S]$ の次数が高々 k であるとき，S を k-独立集合と呼ぶ．

頂点重み付き単純無向グラフ $G = (V, E; w)$, $w : V \to \mathbb{R}$ が与えられているとする．ここで $\mathbb{R}$ は実数の集合である．頂点集合 $S \subseteq V$ について，S の重みを $w(S) = \sum_{v \in S} w(v)$ と定義する．$G[S]$ の重みは S の重みである，すなわち $w(G[S]) = w(S)$ と定義する．最大重み k-独立集合問題は $w(G[S_{\max}])$ が最大となる頂点集合 $S_{\max} \subseteq V$ を求める問題である．

[1] 東京大学
The University of Tokyo
[2] 京都大学
Kyoto University
[a)] denzumi@mist.i.u-tokyo.ac.jp
[b)] jkawahara@i.kyoto-u.ac.jp

2.2 色部分集合と多分決定グラフ

自然数 m と台集合 $U = \{x_1, \ldots, x_m\}$ を固定する．U の要素には $x_1 < x_2 < \cdots < x_m$ の順が付けられていると仮定する．自然数 c に対し，$\vec{D} = (D_0, D_1 \ldots, D_{c-1})$, $D_i \subseteq U$, $D_i \cap D_j = \emptyset\ (i \neq j)$, $\bigcup_{i=0,1,\ldots,c-1} D_i = U$ とすると，$\vec{D}$ を，U を台とする c-色部分集合と呼ぶ．

零抑制型二分決定グラフ（Zero-suppressed binary decision diagram，以下 ZDD） [7] は集合族をコンパクトに表すデータ構造である．c 分決定グラフ（以下，c-DD） [6] は，c-色部分集合を表すデータ構造である．2-DD において，2-DD が表す 2-色部分集合族に含まれる各集合 $\vec{D} = (D_0, D_1)$ について，D_1 によって，（色のない）集合を表すことにすると，2-DD は ZDD の定義とほどんど同じになる．本稿では ZDD の定義を省略し，c-DD の定義のみ説明する．$U = \{x_1, \ldots, x_m\}$ を台とする c-DD は以下の性質を持つ非巡回有向グラフ $H = (N, A)$ である．

- H は出次数が 0 の節点を（原則）ちょうど 2 個持つ．それぞれを 0-終端と 1-終端と呼び，$\bot, \top$ と表記する．
- H は入次数が 0 の節点をちょうど 1 個持つ．その節点を根節点と呼ぶ．
- N に含まれる非終端節点は，ある自然数 i についてラベル x_i を持つ．非終端節点 ν のラベルを $\ell(\nu)$ と表記する．
- N に含まれる非終端節点は出次数がちょうど c であり，各節点から i-枝 $(i = 0, 1, \ldots, c-1)$ と呼ばれる c 本の枝が出る．
- $\nu, \nu' \in N$ について，ν, ν' が非終端節点であるとする．ν から ν' への枝が存在するとき，$\ell(\nu) < \ell(\nu')$ である．

$H = (N, A)$ は以下の通りに c-色部分集合の族を表現すると解釈する．根節点から 1-終端節点までの 1 本の有向パスを考え，これを $P = \langle \nu_1, a_1, \nu_2, a_2, \ldots, a_{m-1}, \top \rangle$ とする．ここで $\nu_i \in N$, $a_i \in A$ である．このパスは c-色部分集合 $\vec{D} = (D_0, D_1, \ldots, D_{c-1})$ を表現すると考える．ここで $i = 0, 1, \ldots, c-1$ に対し，$D_i = \{\ell(\nu_j) \mid a_j$は i-枝 $\}$ とする．H の根節点から 1-終端節点までのすべての有向パスを考え，それぞれが表現する c-色部分集合を集めて得られる族を，H が表現する c-色部分集合族であると解釈する．

ZDD によって（色のない）集合族を表現できる．$G = (V, E)$ を固定すると，部分辺集合 $E' \subseteq E$ によって G の部分グラフ (V', E') を表すことができる．ここで，$V' = \{v \mid \{v, w\} \in E'\}$ である．部分辺集合の族を部分グラフの族とみなすことにより，部分グラフの族を ZDD で表現できる．

3. 提案手法

3.1 提案手法の概要

与えられた頂点重み付きグラフ $G = (V, E; w)$ に対し，すべての G の k-独立集合からなる集合族を ZDD として構築できれば，その ZDD から最大重みの k-独立集合を求めるのは容易である [5]．k-独立集合は頂点集合 V の部分集合であるため，この ZDD は V を台とするのが自然である．次数が高々 k である G の部分グラフ全体の集合を表す ZDD は，フロンティア法と呼ばれる手法により構築可能である [3]．しかしながら，この手法を k-独立集合族の ZDD 構築に用いるためには，2 つの点を考慮する必要がある．第 1 の点として，フロンティア法で構築可能な部分グラフ集合は，辺集合 E を台とする ZDD であるが，k-独立集合族の ZDD は頂点集合 V を台とする．本研究で提案する手法は，$E \cup V$ を台とする DD を扱うことでこの違いに対処する．第 2 の点として，フロンティア法で構築可能な部分グラフ集合は，全頂点の次数が k 以下である，任意の頂点集合が誘導する誘導部分グラフを含んでいるが，誘導部分グラフではないものも含んでいる．そこで本研究では，文献 [4] で提案されている禁止誘導部分グラフ集合族の DD を構築するためのアイデアを用いて，この問題に対処する．

提案手法の概要は以下の通りである．

(1) 次数が高々 k である G の部分グラフ全体の集合を，$E \cup V$ を台とする ZDD $\mathcal{Z}_1$ で表す（この時点ではある頂点集合が誘導する誘導部分グラフとは限らない）．
(2) 頂点集合の誘導部分グラフとなるもののみ全体を表す集合を，$E \cup V$ を台とする 3-DD $\mathcal{Z}_2$ で表す．
(3) $\mathcal{Z}_2$ から ZDD $\mathcal{Z}_3$ に変換する．$\mathcal{Z}_3$ が所望の k-独立集合族を表す．

3.2 DD の変数順

一般に，DD の変数順（台集合の変数が DD 上で現れる順）は，DD の構築前に決めなければならない．提案手法で扱う DD は辺変数と頂点変数の両方を含むため，その出現順を適切に決めなければ，計算時間や DD の大きさが大幅に悪化する [5]．

本稿で提案する変数順の決め方では，最初に辺の変数順を任意に決める．$E = \{e_1, \ldots, e_n\}$ とすると，$e_1 < e_2 < \cdots < e_n$ として一般性を失わない．次に，各頂点変数 $v \in V$ について，頂点変数の位置を以下の通りに決める．v に接続する辺を $e_{f(1)}, e_{f(2)}, \ldots, e_{f(h)}$ $(f(1) < f(2) < \cdots < f(h))$ とすると，v を $e_{f(h)}$ の直後（すなわち $e_{f(h)} < v < e_{f(h)+1}$）に置く．ある辺変数 $e \in E$ の直後に置かれる頂点変数が 2 つ以上存在する場合（実際は高々 2 つしか存在しない），それらの頂点変数の順は任意に定める．以降で紹介するアルゴリズムは，辺変数と頂点変数がこの変数順に従うことを仮定する．

3.3 フロンティア法

提案手法の概要で述べた (1) と (2) は，フロンティア法 [3] と呼ばれる枠組みを用いて，DD を構築する．本小

節ではフロンティア法一般の枠組みの概要について述べ，(1) と (2) については次小節以降で述べる．詳細は文献 [3] を参照されたい．

フロンティア法では，所望の性質を満たす DD を根節点から終端節点の方向に向かって作成する．本小節で作成する DD の台集合を $U = \{x_1, \ldots, x_m\}$ とし，変数順は $x_1 < x_2 < \ldots < x_m$ とする．（本提案手法で作成する DD の台集合は実際は $E \cup V$ であることに注意されたい．）最初にラベル x_1 を持つ根節点とその節点から出る i-枝（$i = 0, \ldots, c-1$）を作成し，i-枝の先に，ラベル x_2 を持つ節点を作成する．以下，ラベル x_j を持つ節点について，その節点から出る i-枝（$i = 0, \ldots, c-1$）と，その先のラベル x_{j+1} を持つ節点の作成を，$j = 1, \ldots, m$ の順に，幅優先的に行う．作成された節点には，問題に依存した何らかの情報を記憶する．その情報を用いて，ある節点が所望の条件を満たすかを判定し，満たさないことが確定した時点で，その節点の代わりに終端節点 $\bot$ を接続させる（枝刈と呼ばれる）．$j = 1, \ldots, m$ まで進み，所望の条件を満たすことが確定した場合は，終端節点 $\top$ を接続させる．2 つの節点について，ラベルが等しく，記憶している情報が等しい場合は，2 つの節点の併合を行う（節点共有と呼ばれる）．

3.4 頂点次数が k 以下の部分グラフ集合を表す ZDD 構築

本小節では概要の (1)，すなわち頂点次数が k 以下の部分グラフ集合を表す ZDD の構築について述べる．頂点次数が k 以下の部分グラフ集合を表す ZDD 構築は，文献 [3] で提案されている．この手法では，節点に記憶させる情報は，フロンティアと呼ばれる頂点集合の各頂点の次数である．ここで，ラベルが x である節点のフロンティアは，変数 x の直前にある辺変数を e_i とすると，$F_i = (\cup_{j=1,\ldots,i} e_j) \cap (\cup_{j=i+1,\ldots,m} e_j)$ で定義される．ただし，$x = e_1$ のときは，$F_0 = \emptyset$ がフロンティアである．本研究で提案する手法で異なる点は，$E \cup V$ を台集合としている点のみである．頂点変数をラベルに持つ節点を作成する際は，単に親の節点の情報をコピーすればよい．頂点変数をラベルに持つ節点の作成時に枝刈は行わない．節点共有は前小節で述べた方法で行う．この方法で得られた ZDD は，集合族 $\{E' \mid (V, E')$ は全頂点の次数が k 以下 $\} \bowtie \{V' \mid V' \subseteq V\}$ を表す．ここで，$\mathcal{A} \bowtie \mathcal{B} = \{A \cup B \mid A \in \mathcal{A}, B \in \mathcal{B}\}$ である．部分グラフとして矛盾するもの（辺 $e = \{y, z\}$ が含まれるが，y や z が含まれない等）が含まれていてもよい．

本研究では，文献 [3] にはない，以下の工夫を行う．各節点作成時に，フロンティア上のある頂点について，記憶されている次数と，未処理の辺（すなわち，節点のラベルの変数より後にある辺変数）の本数の合計が k 以下となるとき，その頂点については，次数が $k+1$ を超えることがないことが確定するため，次数を記憶しなくてよい．その頂点の次数の情報を消去することで，節点の共有が行われやすくなり，構築した ZDD の節点数が減ることが期待される．

3.5 頂点集合の誘導部分グラフ全体を表す 3-DD 構築

本小節では最初に，概要の (2)，すなわち次数が高々 k となる制約を無視して，ある頂点集合の誘導部分グラフ全体を表す 3-DD を構築する手法を述べる．その次に，次数が高々 k という制約を課す方法について述べる．本小節で提案する手法は，文献 [4] で提案されている禁止誘導部分グラフ集合族の DD を構築するためのアイデアを用いている．文献 [4] と異なる点は，本研究の手法では，頂点集合の誘導部分グラフとなるもののみを残すことに特化した手法であることと，辺変数と頂点変数の両方が存在して，それぞれに異なる意味を持たせていることである．

構築する 3-DD は，$E \cup V$ を台とする 3-色部分集合の族であり，各要素 $\vec{D} = (D_0, D_1, D_2)$ は以下の性質を満たす G の部分グラフ $G' = (V', E')$，$V' \subseteq V$，$E' \subseteq E$ を表す．

- $D_i \cap D_j = \emptyset$ $(i \neq j)$
- $D_0 \cup D_1 \cup D_2 = E \cup V$
- $D_2 \cap V = V'$，$D_1 \cap V = \emptyset$，$D_0 \cap V = V \setminus V'$
- $D_2 \cap E = E'$，$(D_0 \cup D_1) \cap E = E \setminus E'$

非形式的に言うと，3-DD の節点は以下の意味を持つ．節点 ν が辺変数のラベル e_i を持つとき，ν から出る 0-枝と 1-枝は e_i を部分グラフに含めないことを意味し，2-枝は e_i を部分グラフに含めることを意味する．0-枝と 1-枝の使い分けは後述する．ν が頂点変数のラベル v を持つとき，ν から出る 0-枝は v を部分グラフに含めないことを意味し，1-枝は常に $\bot$ を指し，2-枝は v を部分グラフに含めることを意味する．

この 3-DD をフロンティア法の枠組みで構築する手法を述べる．節点に記憶させる情報は，フロンティア上の各頂点に対し，3 つの値 *UnDet*（未確定），*Used*（その頂点を部分グラフの頂点として使用することを確定），*NotUsed*（その頂点を部分グラフの頂点として使用しないことを確定）のいずれかである．初期値はすべての頂点が *UnDet* である．

節点 ν が辺変数のラベル $e_i = \{u, x\}$ を持つとき，j-枝（$j = 0, 1, 2$）の先の節点をどのように作成するかについて述べる．以下の 4 通りに場合分けをする．(i) u と x が両方とも *Used* であるとき，誘導部分グラフの条件から，$e_i = \{u, x\}$ は誘導部分グラフ（の辺集合）に含められなければならない．従って，$j = 0, 1$ のときは枝刈を行う．すなわち j-枝の先の節点を $\bot$ にする．(ii) u または x の少なくとも一方が *NotUsed* であるとき，$e_i = \{u, x\}$ は誘導部分グラフ（の辺集合）に含められてはいけない．従って，$j = 2$ のときは枝刈を行う．このとき，1-枝は使用せずに，

0-枝で，e_i を使用しないことを表現する．このため $j=1$ のときも枝刈を行う．(iii) 条件 (i),(ii) が成り立たず，かつ一方の頂点（u として一般性を失わない）が *Used* であるときを考える．$j=2$ のときは e_i を使用するという意味になる．このとき，x も使用しなければならないので，x を *Used* に設定する．e_i を使用しないことは，この場合も 0-枝で表現することにすると，$j=1$ のときに枝刈を行い，$j=0$ のときは x を *NotUsed* に設定する（x が使用されると，u と x が使用されているにも関わらず e_i が使用されないことになり，誘導部分グラフの条件に違反するため）．(iv) 条件 (i),(ii),(iii) が成り立たない場合，u と x は両方とも *UnDet* である．$j=2$ のときは e_i を使用するという意味になるため，u と x の両方を *Used* に設定する．e_i を使用しないとき，u と x の少なくとも一方の頂点は使用してはいけない．一方を使用しないことにすると，他方は使用してもしなくてもよい．従って，u を *NotUsed* にして x を *UnDet* にする状況と，x を *NotUsed* にして u を *UnDet* にする状況の 2 通りを考える必要がある．この 2 通りを，0-枝と 1-枝に割当てる．すなわち，$j=0$ のとき，u を *NotUsed* に設定して，$j=1$ のとき，x を *NotUsed* に設定する．これが，0-枝と 1-枝の両方で，e_i を使用しないという意味を持たせる理由である．

節点 ν が頂点変数のラベル v を持つとき，j-枝（$j=0,1,2$）の先の節点をどのように作成するかについて述べる．頂点変数のラベル v を持つ節点を作成する時点で，ν の先祖（根に近い方の節点）に，v に接続する辺変数のラベルがすべて現れている．従って，v は *UnDet*，*Used*，*NotUsed* のいずれかになることが確定し，今後（ν の子孫節点において）変更されることはない．以下の 3 通りに場合分けされる．(i) v が *UnDet* なら，v は使用してもしなくても，誘導部分グラフの条件に違反しない．v を使用しないことを 0-枝で表現することにし，$j=1$ の場合のみ枝刈り行う．(ii) v が *Used* なら，v は使用されなければならないので，$j=0,1$ のときのみ枝刈を行う．(iii) v が *NotUsed* なら，v は使用されてはいけないので，$j=1,2$ のときのみ枝刈を行う．

節点 ν が変数順において最後の変数ラベル（必ず頂点変数である）を持つとき，上記の枝刈が起こらなければ，$\top$ 節点に接続する．以上の方法により，次数が高々 k となる制約を無視して，ある頂点集合の誘導部分グラフ全体を表す 3-DD を構築することができる．

次に，次数が高々 k という制約を課す方法について述べる．次数が高々 k の部分グラフ全体の集合は概要 (1) の $\mathcal{Z}_1$ として構築済みである．この $\mathcal{Z}_1$ と上記の 3-DD を構築する手法を用いて，サブセッティング法 [2] と呼ばれる手法によって 3-DD 構築を行うことで，次数が高々 k という制約を課すことができる．その際，3-DD の 0-枝と 1-枝を，ZDD $\mathcal{Z}_1$ の 0-枝に，3-DD の 2-枝を $\mathcal{Z}_1$ の 1-枝に対応させる．詳しくは文献 [2] を参照されたい．

3.6 k-独立集合族を表す ZDD の構築

本小節では概要の (3) について述べる．前小節で述べた 3-DD $\mathcal{Z}_2$ を，k-独立集合族を表す ZDD に変換する．3-DD $\mathcal{Z}$ を入力とするこの変換関数を $g(\mathcal{Z})$ と表記する．$g(\mathcal{Z})$ を計算する提案手法を述べる．この計算手法は，DD の再帰構造を利用した計算法である [1].

3-DD $\mathcal{Z}$ の i-枝の先の節点を考える．この節点から到達可能な節点と枝の集合は，3-DD をなす．この 3-DD を $\mathcal{Z}_i$ と表記する．

$\mathcal{Z}$ の根節点が頂点変数 v の場合を考える．このとき，前小節の 3-DD の作り方から，$\mathcal{Z}_1$ は $\perp$ のみからなる．0-枝が v を含まないことを，2-枝が v を含むことを意味する．$g(\mathcal{Z})$ の計算，すなわち出力となる ZDD の作成は以下の通りに行う．ラベル v を持つ根節点を作成し，その 0-枝の先が，$\mathcal{Z}_0$ に再帰的に g を適用して得られる ZDD $g(\mathcal{Z}_0)$ の根節点を指すようにし，その 1-枝の先が，$g(\mathcal{Z}_1)$ の根節点を指すようにすればよい．以上により，$g(\mathcal{Z})$ が作成された．

$\mathcal{Z}$ の根節点が辺変数 e の場合を考える．このとき，出力の ZDD が表す集合族に含まれる各集合は e を含まない．従って，3-DD $\mathcal{Z}$ が e を含んでいるものと，含んでいないものの両方について，再帰的に g を適用して，和集合を取ればよい．$g(\mathcal{Z})$ の計算，すなわち出力となる ZDD の作成は以下の通り行う．$g(\mathcal{Z}_0)$ と $g(\mathcal{Z}_1)$ と $g(\mathcal{Z}_2)$ を計算し，それらの ZDD の和集合（すなわち，ZDD によって表される集合族の和集合の集合族を表す ZDD）を計算すればよい [1], [7].

再帰の末尾は $g(\perp)=\perp$，$g(\top)=\top$ である．以上の方法によって，$g(\mathcal{Z})$ が計算できる．$g(\mathcal{Z})$ の計算中で，辺変数をラベルに持つ節点を 1 つも作成していないため，出力の ZDD が表す集合族に含まれる任意の集合は，辺変数を含まない．

4. 実験

前節で設計したアルゴリズムの実装を行った．実装言語は C++ であり，トップダウン ZDD 構築ライブラリとして TdZdd [2] を，ZDD 操作ライブラリとして SAPPOROBDD[*1] を用いている．

整数計画法による手法との比較を行う．整数計画法を用いた k-独立集合の解法では，グラフの各辺 e を使用するかを表す 0/1 変数 x_e と，グラフの各頂点 v を使用するかを表す 0/1 変数 x_v の 2 種類を用意する．各頂点に対し，その頂点に接続する辺を $e_1,\ldots,e_h$ とすると，$\sum_{i=1,\ldots,h} x_{e_i} \le k$ という制約不等式を用意する．この制約不等式は，各頂点

[*1] https://github.com/Shin-ichi-Minato/SAPPOROBDD

の次数が k 以下であることを表す．各辺 $e = \{v, w\}$ に対し，$x_v + x_w - 1 \leq x_e$，$x_e \leq x_v$，$x_e \leq x_w$ という制約不等式を用意する．この制約不等式は，e が不使用の際に，v と w が同時に使用されることを禁止できる．目的関数は $\sum_{v \in V} w(v)x_v$ である．

実験結果を表 1 に示す．IP time が整数計画法による手法の計算時間，DD time が提案手法による計算時間である（単位は秒）．多くの場合で整数計画法による手法の方が高速であるが，いくつかの場合で提案手法の方が高速なときがある．提案手法では，k-独立集合をすべて求めることに相当する計算を行っており，解を 1 つ求める整数計画法より多くの（1 つ解を求めるだけの場合は不要な）計算を行っている．

5. おわりに

k-独立集合族を表す ZDD を構築するアルゴリズムを提案した．ZDD のよく知られた性質を用いることで，重み最大の k-独立集合族を求めるだけでなく，与えられたグラフのすべての k-独立集合の列挙や数え上げ，一様ランダムサンプリング，様々な条件による抽出，組合せ遷移 [9] 等が可能となる．

今後の課題として，他手法との比較や，実際の DAG 型ブロックチェーンのデータに対する適用等が挙げられる．

謝辞 本研究の 3, 5 節は JSPS 科研費 JP19H01103, JP20H05794 の助成を，1, 4 節は京都大学とトヨタ自動車の共同研究プロジェクト「モビリティ基盤数理の研究」の支援を受けている．

参考文献

[1] Bryant, R. E.: Graph-based Algorithms for Boolean Function Manipulation, *IEEE Transactions on Computers*, Vol. C-35, No. 8, pp. 677–691 (online), DOI: 10.1109/TC.1986.1676819 (1986).
[2] Iwashita, H. and Minato, S.: Efficient top-down ZDD construction techniques using recursive specifications, *TCS Technical Reports*, Vol. TCS-TR-A-13-69 (2013).
[3] Kawahara, J., Inoue, T., Iwashita, H. and Minato, S.: Frontier-based Search for Enumerating All Constrained Subgraphs with Compressed Representation, *IEICE Trans. Inf. Syst.*, Vol. E100-A, No. 9, pp. 1773–1784 (2017).
[4] Kawahara, J., Saitoh, T., Suzuki, H. and Yoshinaka, R.: Colorful Frontier-based Search: Implicit Enumeration of Chordal and Interval Subgraphs, *In Proceedings of the Special Event on Analysis of Experimental Algorithms (SEA^2 2019)*, Vol. 11544, pp. 125–141 (online), DOI: 10.1007/978-3-030-34029-2_9 (2019).
[5] Knuth, D. E.: *The art of computer programming, Vol. 4A, Combinatorial algorithms, Part 1*, Addison-Wesley (2011).
[6] Miller, D.: Multiple-valued logic design tools, *Proc. of the 23rd International Symposium on Multiple-Valued Logic*, pp. 2–11 (online), DOI: 10.1109/ISMVL.1993.289589 (1993).
[7] Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems, *Proc. of the 30th ACM/IEEE design automation conference*, pp. 272–277 (online), DOI: 10.1145/157485.164890 (1993).
[8] Sompolinsky, Y., Wyborski, S. and Zohar, A.: PHANTOM and GHOSTDAG: A Scalable Generalization of Nakamoto Consensus (2018). `https://ia.cr/2018/104`.
[9] 伊藤健洋, 川原純, 宋剛秀, 鈴木顕, 照山順一 and 戸田貴久: ZDD を用いた組合せ遷移ソルバーについての考察, *2021 年度冬の LA シンポジウム* (2022).

表 1 グリッドグラフに対する計算結果
Table 1 Computation results for grid graphs

Graph	$\|V\|$	$\|E\|$	k	IP time	DD time	#nodes	#graphs
grid4x4	16	24	1	0.043096065521240234	0.02s	153	6547
			2	0.04185032844543457	0.02s	198	30824
			3	0.035677433013916016	0.02s	56	58640
grid5x5	25	40	1	0.051009416580200195	0.02s	551	720417
			2	0.12733697891235352	0.02s	891	8402216
			3	0.038613319396972656	0.02s	213	26536192
grid6x6	36	60	1	0.059740543365478516	0.02s	1792	216173426
			2	0.0581357479095459	0.02s	3409	7664347268
			3	0.04099774360656738	0.03s	760	45851039232
grid7x7	49	84	1	0.07511067390441895	0.03s	5446	177509416175
			2	0.8129837512969971	0.05s	12169	23371379782671
			3	0.07259893417358398	0.07s	2572	302758305892480
grid8x8	64	112	1	0.08470296859741211	0.04s	15763	398239490006383
			2	1.6939547061920166	0.15s	41372	238225926162821893
			3	0.17508411407470703	0.24s	8336	7638804476736307712
grid9x9	81	144	1	1.2205414772033691	0.09s	43959	2441922679051541299
			2	2.1343226432800293	0.52s	134612	8118262028301675826132
			3	0.21929407119750977	0.99s	26103	73643772473131216256768 0
grid10x10	100	180	1	0.10888361930847168	0.21s	119087	40923800121824894177005
			2	7.245657920837402	1.90s	424671	92488756323597486010874 6534
			3	1.337935447692871	4.16s	79496	27128763919599722189685 5543808
grid11x11	121	220	1	0.396209716796875	0.59s	315163	18744342234051395116370 99884
			2	15.516634702682495	6.78s	1307767	35226184511279053594191 7078458268
			3	3.224172592163086	16.16s	236658	38186243086867254456636161 3406502912