最小ブロック転送問題に対する $(2-\varepsilon)$ 近似アルゴリズム

八木田 剛[1,a)]　朝廣 雄一[2,b)]　宮野 英次[1,c)]

概要：本論文では，以下のようなグラフ分割問題の一つである最小ブロック転送問題を扱う：有向非巡回グラフ（directe acyclic graph, DAG と略記する）が与えられたとき，根から葉へのパスに沿ってその構造を見ていくときに，外部有向辺数の最大数を最小化するような，DAG のノードからサイズが高々 B の複数ブロックへの分割を見つけることを目的とする．ここでの外部有向辺とは異なる 2 つのブロックをつなぐ有向辺と定義する．本問題に関しては，ブロックのサイズが $B=2$ かつ与えられた DAG の高さが 3 であるときでさえも NP-困難であり，更に $P=NP$ ではないと仮定したときに，任意の正の数 ε_0 について，$B=2$ のときに $(3/2-\varepsilon_0)$ 近似アルゴリズムが設計できないことが知られている．本論文では，$B=2$ のとき，ある正の数 ε について，$(2-\varepsilon)$ 近似アルゴリズムを設計する．

キーワード：最小ブロック転送問題，有向非巡回グラフ，近似アルゴリズム

$(2-\varepsilon)$-Approximation Algorithm for the Minimum Block Transfer Problem

Tsuyoshi Yagita[1,a)]　Yuichi Asahiro[2,b)]　Eiji Miyano[1,c)]

Abstract: In this paper we consider the following variant of clustering problems of graphs, called the Minimum Block Transfer problem: Given a directed acyclic graph (DAG for short), the objective is to find a mapping of its nodes into blocks of size at most B that minimizes the maximum number of external arcs during traversals of the acyclic structure by following paths from the roots to the leaves. An external arc is defined as an arc connecting two distinct blocks. It is known that the problem is NP-hard even if $B=2$ and the height of the DAG is three, and furthermore, there is no $(3/2-\varepsilon_0)$ factor approximation algorithm for $B=2$ and any positive ε_0 unless P = NP. In this paper, when $B=2$, we design a $(2-\varepsilon)$ factor approximation algorithm for a small postive ε.

Keywords: Minimum block transfer problem, directed acyclic graph, approximation algorithm

1. はじめに

1.1 最小ブロック転送問題

有向非巡回グラフ（Directed Acyclic Graph, DAG）は persistent(multiversion) B-trees や ordered binary-decision diagrams(OBDDs) などの重要なデータ構造を含み，様々な種類の情報のモデル化によく用いられている．例えば，順番の制約を持つタスクのスケジューリングや，ネットワークにおけるあるノードから異なるノードへの到達可能性も DAG の構造を用いて表現することが可能である．また，バイオインフォマティクスの分野においても，ゲノムマッピングのデータに対する問い合わせやデータ保管に関して，DAG は一つの優れた表現方法として知られており，Gene Ontology DAG は遺伝子グループの可視化に用いられている [9], [10].

近年では，インターネット上の Web ネットワークや，Twitter や Facebook などのソーシャルネットワーク，ゲノムデータベースなどに見られるように巨大スケールのグラ

[1] 九州工業大学情報工学部 〒 802-8502 飯塚市川津 680-4
[2] 九州産業大学情報科学部 〒 813-8503 福岡市東区松香台 2-3-1
[a)] yagita@theory.ces.kyutech.ac.jp
[b)] asahiro@is.kyusan-u.ac.jp
[c)] miyano@ces.kyutech.ac.jp

フが多く見うけられるようになってきており，そのスケールは日々急速に発達している．しかしそのような非常に巨大なグラフを計算機上で扱う際に，計算機内部のメモリにはグラフのわずかな断片しか保管することが出来ない．よって多くの場合，メインメモリなどの内部メモリと，ディスクなどの外部メモリ間でのデータの転送が計算時間のボトルネックとなる．本論文では，Aggarwal と Vitter[1] によって提案された 2 つのレベルによる I/O モデル上での処理を行うグラフを考える．2 つのレベルとは，限られた容量を持つ内部メモリと，サイズが B の連続するブロックに分割することの出来る，任意の大きさを持つ外部メモリによる階層構造からなる．ここで，外部メモリへの問い合わせや修正はそれぞれ “ブロック転送” と呼ばれるものとし，外部メモリから内部メモリへ B 個のオブジェクトによる 1 つのブロックを転送すると考えるものとする．内部メモリへのアクセス構造は通常，ポインタにより結び付けられる，動的に割り当てられたオブジェクトの集合によって構築され，要は幅の広くない高さの大きいデータ構造が好ましいとされるが，外部メモリのデータ構造としては，ブロック転送回数をできるだけ少なくするため幅が広く高さが小さいものでなければならない．例えば，外部メモリに対する最も基礎的な探索木の 1 つである B-tree の例を見ると，高さは 3 や 4 などのように小さく，幅は大きいという構造になっていることが確認できる．ならば，外部メモリに対する内部メモリを考慮して設計された DAG の構造を用いる際に，以下の疑問が自然と生まれる：“多くのノードにアクセスする可能性があるのに対し，ほんの数ブロックが外部メモリから内部メモリへ転送されるが，果たして DAG のどのノードがディスクのどのブロックにマッピングされるのか？”

本論文では，グラフに関する以下の**サイズ B の最小ブロック転送問題**（Minimum Block Transfers with B, MBT(B)）と呼ばれる問題を考える [8]．この問題は，有向非巡回グラフが入力として与えられ，根から葉へのパスを辿る横断を考えたときの外部有向辺の最大数を最小化するように，グラフのノードからサイズが高々 B であるブロックへの分割を見つけることを目的とする．外部有向辺とは，異なるブロック同士を接続する有向辺のことを意味する．従って，外部有向辺の数とは 2 つの異なるブロックを接続する有向辺の数によって定義され，これはすなわちブロック転送の回数を表している．

1.2 既存研究

MBT(B) は最初に Gil と Itai によって提案された [8]．当初は入力を有向非巡回グラフ（以下 DAG）ではなく木グラフのみに限定し，全てのクエリに対して，予想されるブロック転送回数の数を最小化しようとすることが目的であった．Gil らは，Δ を全頂点中の最大次数としたとき，n 個の頂点を各サイズが B のブロックに分割する最適な分割を，$O(B \log n)$ の領域と $O(nB^2 \log \Delta)$ 時間で求めることが出来る動的計画法に基づくアルゴリズムを提示した．その後，Alstrup らはより厳密な解析を用いて時間の見積もりを $O(nB^2)$ まで改善し，同条件のもとでより早く動作するが近似となるアルゴリズムを提示した [2]．また，Gil らは [8] にて，木グラフのコンパクトな最適分割を見つける問題は NP-困難であることを証明している（ここでのコンパクトとは，総ブロック数が最小化された状態のことを言う）．Diwan ら [5] と Clark ら [4] はそれぞれ独立に，木グラフにおける入出力の最悪ケースに関する問題を考えている．これはすなわち，ブロック転送回数の最大回数を最小化しようとすることが目的である．木グラフに限定した場合は多項式時間で最適な分割が見つかることが示されており，先程のコンパクトの制約を満たすような問題は NP-困難であることが示されている．加えて，Diwan らは [5] にて，入力を DAG に制限したときのブロック転送最大回数の最小化問題，すなわち本論文で扱う MBT(B) にも言及しており，ボトムアップ方式に分割を行うヒューリスティクアルゴリズムを提示しているが，その理論的な近似保証は同論文では行われていない．しかし，このアルゴリズムは後に Asahiro らにより MBT(B) に対する B 近似アルゴリズムであることが示されており，$B = 2$ の場合には近似率は 2 となる [3]．また，近年，Donovan らは [6] にて，MBT(B) の電子回路設計における応用を示し，さらに，最長路に対して繰返し分割することを基本にした近似アルゴリズムを提案しているが，近似率が 3 となってしまう例が存在する．

Asahiro らは [3] にて，DAG を入力とする MBT(B) の NP-困難性を証明している．さらに，$P = NP$ でないと仮定した時，任意の $\varepsilon_0 > 0$ に対して MBT(2) が $(3/2 - \varepsilon_0)$ 近似困難であることを示している．また，任意の高さをもつ DAG を入力とする MBT(2) に対するアルゴリズムを提示し，その近似率が 3/2 となると主張していた．

1.3 本論文の結果

本論文では，ブロックサイズ B は 2 とした MBT(2) に対する近似可能性について考える．本論文の主要な結果は以下である．

(1) 文献 [3] で示されていた MBT(2) に対するアルゴリズムの近似率は少なくとも 8/5 以上であることを示す．
(2) MBT(2) に対して，近似率が，高さ $h(G)$ が奇数のとき $2 - 2/(h(G)+1)$，偶数のとき $2 - 4/h(G)$ となる近似アルゴリズムを示す．

(1) の結果は，文献 [3] で主張されていた近似率 3/2 の主張が間違っていることを意味する．また，(2) の結果より，これまでの最適な近似率 2 を僅かではあるが改善可能であることを示している．

2. 諸定義

$G=(V,A)$ を単純有向非巡回グラフ（DAG），すなわちサイクルや多重辺，自己ループを含まないようなグラフとする．V や A はそれぞれグラフ G の点集合と辺集号を表す．$d^-(v)$ と $d^+(v)$ はある点 v に対する入次数と出次数をそれぞれ表す．入次数や出次数が 0 である点はそれぞれ根（もしくはソース）や，葉（もしくはシンク）と呼ぶ．また，$N^-(v)=\{u \mid (u,v)\in A\}$，$N^+(v)=\{u \mid (v,u)\in A\}$ とし，$A^-(v)=\{(u,v) \mid u\in N^-(v)\}$，$A^+(v)=\{(v,u) \mid u\in N^+(v)\}$ のように定義する．

点 v_0 から点 v_ℓ までの長さ ℓ の（有向）パス Q は，$i=1,2,\cdots,\ell$ のとき $(v_{i-1},v_i)\in A$ であるような点の列 $\langle v_0,v_1,\cdots,v_\ell\rangle$ で表される．ここで，2 つのパス $Q_1=\langle v_0,\cdots,v_i\rangle$ と $Q_2=\langle v_i,\cdots,v_\ell\rangle$ を考えたとき，2 つのパス Q_1 と Q_2 から構成されるパス $Q=\langle v_1,\cdots,v_i,\cdots,v_\ell\rangle$ は，$Q_1\circ Q_2$ のように表す．根からある点 v への最長パスの長さのことを G における v の深さと呼ぶ．対して，ある点 v から葉までの最長パスの長さのことを G における v の高さと呼ぶ．そして，全ての根に対して高さを考えたときの最大のもののことを DAG G の高さと呼び，$h(G)$ で表すこととする．グラフ $G=(V,A)$ に対して，点と辺の各部分集合 $V'\subseteq V$，$A'\subseteq A$ をそれぞれ考えたとき，$G[V']=(V',A')$ は G の誘導部分グラフを表す．このとき A' の端点は V' に属しているものとする．

あるデータが DAG 構造をもつグラフ G にモデル化され，そのデータがディスクに保管されているがサイズが大きいため全体にアクセスするのが困難なときは，G は細かいサイズの複数ブロックに分割される必要がある（すなわちページのことを指す）．つまり，各ブロックのサイズが高々 B であり，$\mathcal{P}=\{P_1,P_2,\cdots,P_k\}$（但し $P_i\subseteq V$，$i\neq j$ のとき $P_i\cap P_j=\emptyset$，$\bigcup_{i=1}^{k}P_i=V$，$|P_i|\leq B$ を満たす）となるようなブロックに分割される必要がある．従って，G のどの点がメインメモリ上のどのブロックにロードされるかを決める必要がある．ここで，分割された各ブロック内の点の数が，予め決めておいたブロックサイズ B を超過していない場合に，この $\mathcal{P}$ のことを分割と呼ぶことにする．分割 $P\in\mathcal{P}$ に対して $\{u,v\}\subseteq P$ のとき，辺 (u,v) はパックされている，と呼ぶこととする．また，分割 $\mathcal{P}$ のもとで，パス Q 上のブロック転送回数 $bt_{\mathcal{P}}(Q)$ とは，u と v がそれぞれ異なるブロックに属しているときの外部有向辺 (u,v) の総数のことを言い，分割 $\mathcal{P}$ のもとでの DAG G のブロック転送回数 $bt_{\mathcal{P}}(G)$ とは，根から葉への全パス中での最大ブロック転送回数のことを意味することとする．アルゴリズム ALG によって出力された分割を $\mathcal{ALG}$ と表す．もし $bt_{\mathcal{ALG}}(G)$ と表記した場合は，分割 $\mathcal{ALG}$ のもとでのブロック転送最大回数を意味することとする．以下に本論文で取り扱う問題を定式化する．

最小ブロック転送問題 (MBT(B))
入力： DAG 構造 $G=(V,A)$ と整数 B
出力： 外部有向辺の最大数，すなわちブロック転送最大回数が最小になるような点集合の分割 $\mathcal{P}$

本問題に対して，DAG の高さを $h(G)$ としたとき，最適な分割 $\mathcal{OPT}$ は以下の事実に示すような下界を持つことが知られており，近似アルゴリズムの近似精度保証等で大きな役割を持つ．

事実 1 ([3]) DAG G とサイズ B のブロックに対し，$bt_{\mathcal{OPT}}(G)\geq\lfloor h(G)/B\rfloor$ である．

最適アルゴリズム OPT とアルゴリズム ALG の分割によるブロック転送回数をそれぞれ $bt_{\mathcal{OPT}}(G)$，$bt_{\mathcal{ALG}}(G)$ と表記する．また，任意の入力 DAG G に対して，$bt_{\mathcal{ALG}}(G)/bt_{\mathcal{OPT}}(G)\leq\sigma$ を満たすような近似アルゴリズム ALG を σ 近似アルゴリズムということとする．

3. MBT(2) に対する既存のアルゴリズム

本節では，MBT(2) に対する既存のアルゴリズムを述べる．まず，第 1 節でも述べたが，MBT(2) に対する既存の結果を改めて以下に述べる．

事実 2 ([5], [3]) MBT(2) に対して，2 近似アルゴリズムが存在する．

また [3] では，高さがそれぞれ高々 1 と 2 の DAG を入力とする MBT(2) に対する最適な分割アルゴリズム HeightOne と HeightTwo が示されている．まず以下に HeightOne を示す．

Algorithm HeightOne
入力: 高さ $h(G)$ が 1 である DAG $G=(V,A)$
出力: 分割 $\mathcal{P}$
Step 1. G 中の全ての連結成分を見つける．連結成分が k 個あるとしたとき，それらの各連結成分を $C_1,\cdots,C_k$ とする．
Step 2. $|C_i|\leq B$（但し $i=1,\cdots,k$）ならば，$\mathcal{P}=\{C_1,\cdots,C_k\}$ を出力する．そうでなければ $\mathcal{P}=\{\{v\} \mid v\in V\}$ を出力する．

$|C_i|=2$ となるような各連結成分 C_i は 1 本の辺のことである．すなわち MBT(2) に対する HeightOne は，孤立辺をパックしようとするアルゴリズムである．また，HeightOne は MBT(2) のみならず MBT(B) に対しても最適である [3]．

次に，以下に HeightTwo を示す．

Algorithm HeightTwo
入力: 高さ $h(G)$ が 2 である DAG $G=(V,A)$
出力: 分割 $\mathcal{P}$
Step 1. 以下の規則に基づき，G を 2-CNF f へと変形する：
Step 1-1: $v_i \in V_1$ の各ノードに対し変数 x_i を割り当てる．
Step 1-2: ノード $v_i \in V_1$ によって形成される以下の節を生成する．
Rule (i): $d^-(v_i) \geq 2$ ならば $(\overline{x_i})$
Rule (ii): $d^+(v_i) \geq 2$ ならば (x_i)
Step 1-3: 各組 $v_i, v_j \in V_1$ に対し以下の節を生成する．
Rule (iii): $d^-(v_i) = d^-(v_j) = 1$ であり，かつ両弧が (u,v_i), (u,v_j) となるような u が V_0 に存在するならば $(\overline{x_i} \vee \overline{x_j})$
Rule (iv): $d^+(v_i) = d^+(v_j) = 1$ であり，かつ両弧が (v_i,u), (v_j,u) となるような u が V_2 に存在するならば $(x_i \vee x_j)$
Step 1-4: Steps 1-2 と 1-3 で生成された節を $\wedge$(かつ) でつなぎ，f を生成する．
Step 2. f を 2-SAT の多項式時間アルゴリズム [7] で解く．
Step 3. もし f が充足不可である場合，分割 $\mathcal{P} = \{\{v\} \mid v \in V\}$ を出力する．f が充足可能であるならば，Step 2 で得られた充足可能となる $true, false$ の組み合わせに基づき，以下のような分割 $\mathcal{P}$ を出力する：
各変数 x_i に対し，
- $x_i = true$ ならば $\mathcal{P}$ に $\{v_i\} \cup N^-(v)$ を追加．
- $x_i = false$ ならば，$\mathcal{P}$ に $\{v_i\} \cup N^+(v)$ を追加．

以上の操作で分割されていない各ノード v に関しては，$\{v\}$ として各頂点個々に分割して $\mathcal{P}$ に追加する．

事実 3 ([3]) HeightOne と HeightTwo はそれぞれ高さ 1 と 2 の DAG を入力とする MBT(2) に対する最適アルゴリズムであり，$O(|V|+|A|)$ 時間で動作する．

さらに Asahiro らは [3] にて，それぞれ高さが 3 と 4 の DAG を入力としたときのアルゴリズム，HeightThree と HeightFour も示している．以下にその詳細を述べる．

Algorithm HeightThree
入力: 高さ $h(G)$ が 3 である DAG $G=(V,A)$
出力: 分割 $\mathcal{P}$
Step 1. $G[V_0 \cup V_1]$ と $G[V_2 \cup V_3]$ に対し HeightOne を適用する．得られた分割を $\mathcal{P}_1$ と $\mathcal{P}_2$ とする．
Step 2. $bt_{\mathcal{P}_1}(G[V_0 \cup V_1]) = 0$ かつ $bt_{\mathcal{P}_2}(G[V_2 \cup V_3]) = 0$ ならば，$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ を出力し動作を停止する．$bt_{\mathcal{P}_2}(G[V_2 \cup V_3]) = 1$ ならば，$\mathcal{P} = \{\{v\} \mid v \in V\}$ を出力し動作を停止する．
Step 3. V_3 に属する点に到達可能な点の集合を R とする．もし $R = V$ ならば，$\mathcal{P} = \{\{v\} \mid v \in V\}$ を出力し動作を停止する．そうでなければ，まず $G[V-R]$ に対し HeightTwo を適用し，次に $G[R \cap (V_0 \cup V_1)]$ と $G[R \cap (V_2 \cup V_3)]$ に対し HeightOne を適用する．そしてそれらの分割を $\mathcal{P}_3$，$\mathcal{P}_4$，$\mathcal{P}_5$ としたとき，それらの分割を取得し，$\mathcal{P}_3 \cup \mathcal{P}_4 \cup \mathcal{P}_5$ を出力し動作を停止する．

Algorithm HeightFour
入力: 高さ $h(G)$ が 4 である DAG $G=(V,A)$
出力: 分割 $\mathcal{P}$
Step 1. $G[V_0 \cup V_1]$ に対し HeightOne を適用し，分割 $\mathcal{P}_1$ を取得する．もし $bt_{\mathcal{P}_1}(G[V_0 \cup V_1]) = 0$ ならば，$\mathcal{P} = \mathcal{P}_1 \cup \{\{v\} \mid v \in V\}$ を出力し動作を停止する．
Step 2. $G[V_0 \cup V_1 \cup V_2]$ に対し HeightTwo を適用し，分割 $\mathcal{P}_2$ を取得する．もし $bt_{\mathcal{P}_2}(G[V_0 \cup V_1 \cup V_2]) = 1$ ならば，$\mathcal{P} = \mathcal{P}_2 \cup \{v\} \mid c \in V_3 \cup V_4\}$ を出力し動作を停止する．
Step 3. $\mathcal{P} = \{\{v\} \mid v \in V\}$ を出力し動作を停止する．

事実 4 ([3]) HeightThree と HeightFour はそれぞれ高さ 3 と 4 の DAG を入力とする MBT(2) に対する 3/2 近似アルゴリズムであり，$O(|V|+|A|)$ 時間で動作する．

最後に，[3] にて示された，任意の高さの DAG を入力とする近似アルゴリズム DAGPack を以下に示す．このアルゴリズムでは，上に記した HeightOne から HeightFour までをサブルーチンとして用いている．

Algorithm DAGPack
入力: DAG $G=(V,A)$
出力: 分割 $\mathcal{P}$
Step 1. $k = \lfloor h(G)/4 \rfloor$ として，$W_0 = V_0 \cup \cdots \cup V_4$，$W_1 = V_4 \cup \cdots \cup V_8$，$\cdots$，$W_k = V_4 k \cup \cdots \cup V_{h(G)}$ とする．
Step 2. $0 \leq i \leq k-1$ として，各部分グラフ $G[W_i]$ に対してアルゴリズム HeightFour を適用する．各 $G[W_i]$ で得られた分割をそれぞれ $\mathcal{P}_i$ とする．

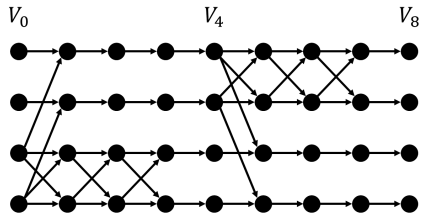

図 1　DAGPack の反例
Fig. 1　Counterexample for DAGPack

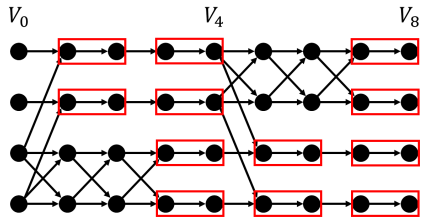

図 2　DAGPack の反例についての最適分割
Fig. 2　Optimal packing for the counterexample

Step 3.　$G[W_k]$ の高さを $h' = h(G) - 4k$ （但し $1 \leq h' \leq 3$）とする．部分グラフ $G[W_k]$ に対し，高さ $h' = 1, 2, 3$ に応じて HeightOne, HeightTwo, HeightThree のいずれかを適用する．得られた分割を $\mathcal{P}_k$ とする．

Step 4.　$(\mathcal{P}_1 - \{\{v\} \mid v \in V_4\}) \cup (\mathcal{P}_2 - \{\{v\} \mid v \in V_8\}) \cup \cdots \cup \mathcal{P}_k$ を出力し動作を停止する．

主張 1 ([3])　DAGPack は MBT(2) に対する 3/2 近似アルゴリズムである．

Asahiro らは DAGPack の近似率に関して以上の主張を行っているが，この主張は誤りである．その反例を**図 1** に示す．

定理 1　DAGPack の近似率は少なくとも 8/5 である．

証明.　図 1 に示す DAG に対して，アルゴリズム DAGPack は全ての頂点を一つずつパックするが，$G[V_3 \cup V_4]$ と $G[V_3 \cup V_4]$ 中の辺を全てパックすることでブロック転送回数は減らすことが可能である．DAGPack の分割によって得られるブロック転送回数は 8 であるが，**図 2** に示すとおり，最適な分割によるブロック転送回数は 5 である．従って，DAGPack による近似率は少なくとも 8/5 である．　■

DAGPack に誤りがあり主張 1 が間違っていたことから，事実 2 でも示したように，高さ 5 以上の DAG を入力とする MBT(2) に対するこれまでの最良の近似アルゴリズムは文献 [5] で提案されたものであり，その近似率は 2 である．よって，本論文の目的はこの近似率 2 を改善することである．

4.　近似アルゴリズム

本節では MBT(2) に対して，近似率が $2 - \varepsilon$ となる，すなわち本質的に 2 よりも小さい近似率を持つ近似アルゴリズムを示す．まずそれに伴い，既存の HeightTwo をわずかに改善した最適アルゴリズム HeightTwo*を以下に示す．

Algorithm HeightTwo*
入力:　高さが 2 である DAG $G = (V, A)$
出力:　分割 $\mathcal{P}$

Step 1.　以下の規則に基づき，G を 2-CNF f へと変形する：

Step 1-1:　$v_i \in V_1$ の各ノードに対し変数 x_i を割り当てる．

Step 1-2:　ノード $v_i \in V_1$ によって形成される以下の節を生成する．

Rule (i):　$d^-(v_i) \geq 2$ ならば $(\overline{x_i})$

Rule (ii):　$d^+(v_i) \geq 2$ ならば (x_i)

Step 1-3:　各組 $v_i, v_j \in V_1$ に対し以下の節を生成する．

Rule (iii):　$d^-(v_i) = d^-(v_j) = 1$ であり，かつ両弧が (u, v_i), (u, v_j) となるような u が V_0 に存在するならば $(\overline{x_i} \vee \overline{x_j})$

Rule (iv):　$d^+(v_i) = d^+(v_j) = 1$ であり，かつ両弧が (v_i, u), (v_j, u) となるような u が V_2 に存在するならば $(x_i \vee x_j)$

Step 1-4:　各 $v_i \in V_1$ に対し，以下の節を生成する．

- $d^-(v_i) \leq 1$ かつ $d^+(v_i) \leq 1$ ならば $(x_i \vee \overline{x_i})$

Step 1-5:　Step 1-2,Step 1-3 および Step 1-4 で生成された節を $\wedge$(かつ) でつなぎ，f を生成する．

Step 2.　f を 2-SAT の多項式時間アルゴリズム [7] で解く．

Step 3.　もし f が充足不可である場合，分割 $\mathcal{P} = \{\{v\} \mid v \in V\}$ を出力する．f が充足可能であるならば，Step 2 で得られた充足可能となる $true, false$ の組み合わせに基づき，以下のような分割 $\mathcal{P}$ を出力する：
各変数 x_i に対し，

- $x_i = true$ ならば $\mathcal{P}$ に $\{v_i\} \cup N^-(v)$ を追加．
- $x_i = false$ ならば，$\mathcal{P}$ に $\{v_i\} \cup N^+(v)$ を追加．

以上の操作で分割されていない各ノード v に関しては，$\{v\}$ として各頂点個々に分割して $\mathcal{P}$ に追加する．

以前の HeightTwo では，長さが 2 の 3 頂点によるパスが $G[V_0 \cup V_1 \cup V_2]$ に存在したときはクローズを生成せず分割を行わなかったため，そのような構造に対してもク

ローズを生成するような **Step 1-4** を追加している．なお，$(x_i \vee \overline{x_i})$ が追加されるのみであるので，f，f' を 2-CNF とすると，$f = (x_i \vee \overline{x_i}) \cdot f' = f'$ となり，HeightTwo*の最適性は満たされる．

以降 HeightOne や HeightTwo*がそれぞれブロック転送回数を 0 もしくは 1 に出来た状態のことを *Yes*，出来なかった状態のことを *No* と呼ぶこととする．アルゴリズムは以下である．

Algorithm NewDAGPack
入力: DAG $G = (V, A)$
出力: 分割 $\mathcal{P}$
Step 0. $\mathcal{P}$，$S_1, \cdots, S_6$ を全て $\emptyset$ とする．
Step 1.（高さが奇数の場合） V_1 内の高さ，$V_{h(G)-1}$ 内の深さがそれぞれ $h(G)-2$ 以下になる点の集合を S_1，S_2 とする．$G[(V_0 \cup V_1) - S_1]$ と $G[(V_{h(G)-1} \cup V_{h(G)}) - S_2]$ に HeightOne を適用し，得られた分割を $\mathcal{P}_1$，$\mathcal{P}_2$ とする．
◯ $h(G) = 3$ **のとき：** HeightOne の適用結果の組み合わせにより以下の場合分けを行う：
- *Yes*-*Yes* 以外のとき：$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ とする．
- *Yes*-*Yes* のとき：$\mathcal{P}_1$ 内の辺 (p, q) に対応するパック $\{p, q \mid p \in V_0, q \in V_1\}$ の q および $\mathcal{P}_2$ 内の辺 (r, s) に対応するパック $\{r, s \mid r \in V_2, s \in V_3\}$ の r を S_3 に追加する．
  また，$\mathcal{P}_1$ 内の点 u に対応するパック $\{u \mid u \in V_1\}$ に対して $\mathcal{P}_1' = \mathcal{P}_1 - \{u\}$ とし，$\mathcal{P}_2$ 内の点 v に対応するパック $\{v \mid v \in V_2\}$ に対して $\mathcal{P}_2' = \mathcal{P}_2 - \{v\}$ とする．
  $G[(V_1 \cup V_2) - S_3]$ に HeightOne を適用し，得られた分割を $\mathcal{P}_3$ とする．$\mathcal{P} = \mathcal{P}_1' \cup \mathcal{P}_2' \cup \mathcal{P}_3$ とする．

◯ $h(G) \geq 5$ **のとき：** HeightOne の適用結果の組み合わせにより以下の場合分けを行う：
- *Yes*-*Yes* 以外の時：$\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ とする．
- *Yes*-*Yes* のとき：$\mathcal{P}_1$ 内の辺 (p, q) に対応するパック $\{p, q \mid p \in V_0, q \in V_1\}$ の q を S_3 に，$\mathcal{P}_2$ 内の辺 (r, s) に対応するパック $\{r, s \mid r \in V_{h(G)-1}, s \in V_{h(G)}\}$ の r を S_4 に追加する．
  $v_i \in V_2$ に対して高さが $h(G)-4$ 以下の点を S_3 に，$v_j \in V_{h(G)-2}$ に対して深さが $h(G)-4$ 以下の点を S_4 に全て追加する．
  また，$\mathcal{P}_1$ 内の点 u に対応するパック $\{u \mid u \in V_1\}$ に対して $\mathcal{P}_1' = \mathcal{P}_1 - \{u\}$ とし，$\mathcal{P}_2$ 内の点 v に対応するパック $\{v \mid v \in V_2\}$ に対して $\mathcal{P}_2' = \mathcal{P}_2 - \{v\}$ とする．
  $G[(V_1 \cup V_2) - S_3], G[(V_{h(G)-2} \cup V_{h(G)-1}) - S_4]$ 両者に HeightOne を適用し，得られた分割を $\mathcal{P}_3$，$\mathcal{P}_4$ とする．$\mathcal{P} = \mathcal{P}_1' \cup \mathcal{P}_2' \cup \mathcal{P}_3 \cup \mathcal{P}_4$ とする．
Step 2.（高さが偶数の場合） $v_i \in V_2$，$v_j \in V_{h(G)-2}$ に対して，高さ，深さがそれぞれ $h(G)-3$ 以下の点の集合をそれぞれ S_5, S_6 とする．$G[(V_0 \cup V_1 \cup V_2) - S_5]$, $G[(V_{h(G)-2} \cup V_{h(G)-1} \cup V_{h(G)}) - S_6]$ に対して HeightTwo*を適用する．得られた分割をそれぞれ $\mathcal{P}_5$，$\mathcal{P}_6$ とする．
◯ $h(G) = 4$ **のとき：** HeightTwo*の適用結果が *Yes* となったどちらか片方の分割 $\mathcal{P}_i$（$i \in \{5, 6\}$）を選択し，$\mathcal{P} = \mathcal{P}_i$ とし **Step 3** へ進む．
両方が *No* となれば，どちらか片方の分割 $\mathcal{P}_i$（$i \in \{5, 6\}$）を選択し，$\mathcal{P} = \mathcal{P}_i$ とし **Step 3** へ進む．
◯ $h(G) \geq 6$ **のとき：** $\mathcal{P} = \mathcal{P}_5 \cup \mathcal{P}_6$ とする．
Step 3. $\mathcal{P} = \mathcal{P} \cup \{\{v\} \mid v \in V, v$ はパックされていない点 $\}$ とし，$\mathcal{P}$ を出力して動作を停止する．

定理 2 アルゴリズム NewDAGPack の近似率は，入力 DAG G の高さ $h(G)$ が，(i) $h(G) \geq 3$ である奇数のとき $2 - 2/(h(G)+1)$，(ii) $h(G) = 4$ のとき 4/3，i(ii) $h(G) \geq 6$ である偶数のとき $2 - 4/h(G)$ である．

証明. まず，事実 1 より，高さ $h(G)$ の DAG G を入力とする MBT(2) に対する下界は $bt_{\mathcal{OPT}}(G) \geq \lfloor h(G)/2 \rfloor$ である．これは高さ $h(G)$ が奇数の場合のみ $bt_{\mathcal{OPT}}(G) = \lfloor h(G)/2 \rfloor$ を達成でき，$h(G)$ が偶数の場合は $bt_{\mathcal{OPT}}(G) = h(G)/2$ となることは明らかである．

(i) $h(G) \geq 3$ である奇数のときを考える．

(i-1) **Step 1** にて，長さが $h(G)$ となるパスのみを対象としたうえで，HeightOne の適用結果が *No*-*No* になる場合には，最適解は *No* として分割できなかった部分を除いて最適に分割しようとするので，$bt_{\mathcal{ALG}}(G) = h(G)$，$bt_{\mathcal{OPT}}(G) \geq \lfloor (h(G)-2)/2 \rfloor + 2$ が成り立つ．高さは奇数を仮定しているため，$h(G) = 2k+1$（但し $k = 1, 2, 3, \cdots$）とすると，

$$\frac{bt_{\mathcal{ALG}}(G)}{bt_{\mathcal{OPT}}(G)} \leq \frac{h(G)}{\lfloor \frac{h(G)-2}{2} \rfloor + 2} = \frac{2k+1}{\lfloor \frac{2k-1}{2} \rfloor + 2} = \frac{2k+1}{k+1}$$
$$= 2 - \frac{1}{k+1} = 2 - \frac{2}{h(G)+1}$$

が成り立つ．

(i-2) 次に，*Yes*-*No*，もしくは *No*-*Yes* になる場合には，*No* となる部分で最適解でもブロック転送回数は少なくとも 1 増え，アルゴリズム解では *Yes* となっている部分が存在す

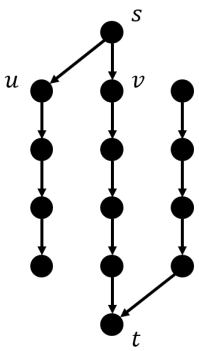

図 3　$bt_{\mathcal{OPT}}(G)=2$ となる DAG G
Fig. 3　DAG G which achieves $bt_{\mathcal{OPT}}(G)=2$

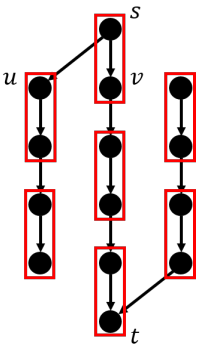

図 4　$bt_{\mathcal{OPT}}(G)=2$ となる分割
Fig. 4　Optimal packing of G

るのでブロック転送回数は少なくとも 1 減る．また，*No-No* のとき同様，最適解は *No* として分割できなかった部分を除いて最適に分割しようとするので，$bt_{\mathcal{ALG}}(G)=h(G)-1$，$bt_{\mathcal{OPT}}(G)\geq\lfloor(h(G)-1)/2\rfloor+1$ が成り立つ．従って，

$$\frac{bt_{\mathcal{ALG}}(G)}{bt_{\mathcal{OPT}}(G)}\leq\frac{h(G)-1}{\lfloor\frac{h(G)-1}{2}\rfloor+1}=\frac{2k}{\lfloor\frac{2k}{2}\rfloor+1}=\frac{2k}{k+1}$$
$$=2-\frac{2}{k+1}=2-\frac{4}{h(G)+1}$$

が成り立つ．

(i-3) 最後に *Yes-Yes* となる場合を考える．例えば**図 3** に示すような DAG に対して，**図 4** のような最適分割を行うことで $bt_{\mathcal{OPT}}(G)=2$ となる例が存在する．アルゴリズムは長さ $h(G)$ のパスに対して *Yes-Yes* とした上で，長さ $h(G)-1$ のパスに対しても，`HeightOne` の最適性からブロック転送回数を 1 減らせるならば分割して減らすため，$bt_{\mathcal{ALG}}(G)=h(G)-2$ が成り立つ．従って $bt_{\mathcal{OPT}}(G)\geq\lfloor h(G)/2\rfloor$ より，

$$\frac{bt_{\mathcal{ALG}}(G)}{bt_{\mathcal{OPT}}(G)}\leq\frac{h(G)-2}{\lfloor\frac{h(G)}{2}\rfloor}=2-\frac{2}{h(G)-1}$$

が成り立つ．

(i-1)，(i-2)，および (i-3) より，$h(G)\geq 3$ である奇数のとき近似率は $2-2/(h(G)+1)$ 以下であることが示された．特に，$h(G)=3$ の場合は *No-No* になる場合にのみ 3/2 近似となることに注意して欲しい．これは最適解が辺集合 $\{(u,v)\mid u\in V_1,v\in V_2\}$ を全てパックし $bt_{\mathcal{OPT}}(G)=2$，アルゴリズム解は $bt_{\mathcal{ALG}}(G)=3$ となる場合などが存在するためである．

(ii) $h(G)=4$ とする．$h(G)=4$ の場合，まず $bt_{\mathcal{OPT}}(G)=h(G)/2=2$ が成り立つ．それは $G[V_0\cup V_1\cup V_2]$ でブロック転送回数を 1 減らし，$G[V_2\cup V_3\cup V_4]$ でブロック転送回数を 1 減らしている場合のみである．アルゴリズム `HeightTwo*`は最適アルゴリズムであるので，最適解が高さ 2 の DAG のブロック転送回数を 1 に出来るならばアルゴリズムもそれを満たす分割を出力する，すなわち *Yes* となる．よって，$bt_{\mathcal{OPT}}(G)=2$ となるような高さ 4 の DAG が入力として与えられた場合，*Yes-Yes* の状態となるが，どちらか片方の *Yes* となった分割を選択し，残りの頂点を 1 つずつパックする分割を行うため，$bt_{\mathcal{ALG}}(G)=3$ が成り立つ．よって $h(G)=4$ のときには 3/2 近似となる．

(iii) 次に，**Step 2** にて高さ $h(G)$ が偶数かつ $h(G)\geq 6$ の場合を考える．

(iii-1) `HeightTwo*`の適用結果が *No-No* となる場合には $bt_{\mathcal{ALG}}(G)=h(G)$，$bt_{\mathcal{OPT}}(G)\geq(h(G)/2)+2$ が成り立つ．従って，

$$\frac{bt_{\mathcal{ALG}}(G)}{bt_{\mathcal{OPT}}(G)}\leq\frac{h(G)}{\frac{h(G)}{2}+2}=2-\frac{8}{h(G)+4}$$

が成り立つ．

(iii-2) 次に，*Yes-No*，もしくは *No-Yes* になる場合には $bt_{\mathcal{ALG}}(G)=h(G)-1$，$bt_{\mathcal{OPT}}(G)\geq(h(G)/2)+1$ が成り立つ．従って，

$$\frac{bt_{\mathcal{ALG}}(G)}{bt_{\mathcal{OPT}}(G)}\leq\frac{h(G)-1}{\frac{h(G)}{2}+1}=2-\frac{6}{h(G)+2}$$

が成り立つ．

(iii-3) 最後に *Yes-Yes* となる場合には，$bt_{\mathcal{ALG}}(G)=h(G)-2$，$bt_{\mathcal{OPT}}(G)\geq h(G)/2$ が成り立つ．従って，

$$\frac{bt_{\mathcal{ALG}}(G)}{bt_{\mathcal{OPT}}(G)}\leq\frac{h(G)-2}{\frac{h(G)}{2}}=2-\frac{4}{h(G)}$$

が成り立つ．

(iii-1)，(iii-2)，および (iii-3) より，入力の DAG の高さ $h(G)$ が偶数かつ $h(G)\geq 6$ のとき，アルゴリズム `NewDAGPack` の近似率が $2-4/h(G)$ 以下となることが示された． ■

また，以上の議論より以下のことが得られる．

系 1　アルゴリズム `NewDAGPack` は，入力 DAG G の高さ $h(G)$ が 3 または 4 のとき，最適な近似アルゴリズムである．

謝辞

本研究の一部は，科学研究費補助金 JP25440018 および JP26330017 による．

参考文献

[1] Alok Aggarwal, Jeffrey Vitter, et al. The input/output complexity of sorting and related problems. *Communications of the ACM*, Vol. 31, No. 9, pp. 1116–1127, 1988.

[2] Stephen Alstrup, Michael A Bender, Erik D Demaine, Martin Farach-Colton, Theis Rauhe, and Mikkel Thorup. Efficient tree layout in a multilevel memory hierarchy. *arXiv preprint cs/0211010*, 2002. Revised in 2003.

[3] Yuichi Asahiro, Tetsuya Furukawa, Keiichi Ikegami, and Eiji Miyano. How to pack directed acyclic graphs into small blocks. In *Italian Conference on Algorithms and Complexity*, pp. 272–283. Springer, 2006.

[4] David R Clark and J Ian Munro. Efficient suffix trees on secondary storage. In *SODA*, Vol. 96, pp. 383–391, 1996.

[5] A. A. Diwan, Sanjeeva Rane, S. Seshadri, and S. Sudarshan. Clustering techniques for minimizing external path length. In *Proceedings of the 22th International Conference on Very Large Data Bases*, pp. 342–353, 1996.

[6] Zola Donovan, Vahan Mkrtchyan, and K Subramani. On clustering without replication in combinatorial circuits. In *Combinatorial Optimization and Applications*, pp. 334–347. Springer, 2015.

[7] S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *j-SIAM-J-COMPUT*, Vol. 5, No. 4, pp. 691–703, 1976.

[8] Joseph Gil and Alon Itai. Packing trees. In *European Symposium on Algorithms*, pp. 113–127, 1995.

[9] Sung Geun Lee, Jae Seong Yang, Wan Seon Lee, Miyoung Shin, and Yang Seok Kim. Extraction of biological contexts and ontological dag structures from gene groups using go term distribution. *Genome Informatics*, Vol. 14, pp. 683–684, 2003.

[10] Mi Zhou and Yan Cui. Geneinfoviz: constructing and visualizing gene relation networks. *In silico biology*, Vol. 4, No. 3, pp. 323–333, 2004.