

有向ハイパーグラフ上での到達可能性判定の高速化

Efficient Algorithm to Reachability Problem on Directed Hypergraph

佐々木 耀一^{*1}

Yoichi Sasaki

木村 圭吾^{*1}

Keigo Kimura

山本 風人^{*1}

Kazeto Yamamoto

岡嶋 穰^{*1}

Yuzuru Okajima

定政 邦彦^{*1}

Kunihiko Sadamasa

^{*1}NEC セキュリティ研究所

Security Research Labs., NEC Corp.

In this paper, we propose new efficient algorithms to reachability problem on directed hypergraph. These algorithms are based on applying ZDD, which is a data structure providing a data compression and the manipulation for a set of combination to transitive closure information.

1. はじめに

有向ハイパーグラフとは、ノード集合の間に成り立つ有向関係を表したグラフ構造である。これは有向グラフとハイパーグラフの両方の性質を持つため、より一般化されたグラフと考えることができ、化合物の化学反応ネットワーク [Özturan 08] や、論理推論における知識ベースなど多くの適用先がある。

有向ハイパーグラフ上での到達可能性判定問題とは、始点ノード集合 S として任意の数のノードと、終点ノード t としてノードが一つ与えられたときに、始点ノード集合 S から終点ノード t へパスが存在するかどうかを判定する問題である。上記の化合物の化学反応ネットワークを例に考えると、任意の化合物が1つ以上与えられ、目的の化合物も1つ与えられたとき、与えられた化合物の中で化学反応を通して目的の化合物が作成可能かを判定する問題にあたる。

この問題に対し、有向ハイパーグラフのノード数を $|V|$ 、エッジ数を $|E|$ としたときに、幅優先探索を行うことで $O(|E|)$ 時間で解くことができる [Ausiello 17]。有向グラフではエッジはノード間の関係を表すので、エッジ数 $|E| = O(|V|^2)$ だが、本研究で扱う有向ハイパーグラフではエッジはノード集合間の関係を表すため、エッジ数 $|E| = O(2^{|V|} \cdot 2^{|V|}) = O(2^{2|V|})$ である。

一方、前処理として推移閉包情報を保持しておく手法が考えられる [Ausiello 17]。前処理として、各ノードに対し、そのノードへパスが到達可能である全ての始点ノード集合をあらかじめ計算し保持しておくことで、定数時間で到達可能性判定を行うことができる。しかし、メモリ使用量としては、 $O(|V| \cdot 2^{|V|})$ 必要である。

メモリ使用量を減らす方法として、各ノードへ到達可能な全ての始点ノード集合の情報ではなく、ハイパーエッジを形成しているノード集合に限定した中で到達可能な全ての始点ノード集合の情報を保持する方法が提案されており、このような情報は簡潔推移閉包 (Succinct Transitive Closure) 情報と呼ばれている [Ausiello 17]。しかし、簡潔推移閉包情報はハイパーエッジが著しく少ないときには有効であるものの、ハイパーエッジが多くなると推移閉包情報のメモリ使用量に近づき、到達可能性判定にかかる計算時間もクエリ依存になってしまい最悪時は幅優先探索と同じ操作になり $O(|E|)$ 時間かかってしまうという問題点がある。

本稿では、推移閉包情報の性質を活かした圧縮を行うこと

により、メモリ使用量を減らしつつ、高速に到達可能性判定を行うことができるアルゴリズムを提案する。提案手法は組合せ集合に特化した圧縮法として ZDD [Minato 93] と呼ばれるデータ構造を利用することにより、圧縮したまま ZDD の高さ ($= O(|V|)$) で到達可能性の判定を行うことを可能にする。

2. 準備

2.1 基本的定義

有向ハイパーグラフ (Directed Hypergraph, DH) とは、 $DH(V, E)$ で定義され、 V はノードの集合、 E はエッジの集合を表す。ノード集合 V に含まれるノードの数を $|V|$ で表し、エッジの数を $|E|$ で表す。また、ノード集合 V のべき集合を $\Pi(V)$ で表す。エッジ $e \in E$ は、あるノード集合 $S = \{s_1, \dots, s_{|S|}\}$ と、 $T = \{t_1, \dots, t_{|T|}\}$ に対して、 $e = (S, T)$ で与えられ、 S から T への方向性を表す。ノード $s \in S$ を入ノード、 $t \in T$ を出ノードと定義し、 $tail(e) = S$ をエッジ e の入ノード集合、 $head(e) = T$ をエッジ e の出ノード集合と定義する。また、 $tail(E)$ で有向ハイパーグラフ $DH(V, E)$ の全てのエッジの入ノード集合の集合を表し、 $head(E)$ で全てのエッジの出ノード集合の集合を表す。

有向ハイパーグラフのエッジとは、単エッジ (single edge) とハイパーエッジ (hyper edge) とに分類される。

単エッジは1つのノードと1つのノードとの間の関係を示す。つまり、単エッジ $e = (S, T)$ は、 S, T 共に1つのノードを要素とするノード集合である。

ハイパーエッジは、1つのノードと複数のノードとの間の関係、または複数のノードと複数のノードとの関係を示す。1つのノードと複数のノードとの間の関係を示すハイパーエッジは、さらに、複数の入ノードおよび1つの出ノードを持つ **B**-ハイパーエッジと、1つの入ノードおよび複数の出ノードを持つ **F**-ハイパーエッジに分類される。また、単エッジ及び **B**-ハイパーエッジのみから構成される有向ハイパーグラフを **B**-有向ハイパーグラフと定義し、単エッジ及び **F**-ハイパーエッジのみから構成されるハイパーグラフを **F**-有向ハイパーグラフと定義する。

始点ノード集合 S から終点ノード t へのパス P とは、以下の条件を満たす文字列 $P = [S, e_1, N_2, e_2, \dots, e_k, N_{k+1}]$ で定義される。

- 各 $j = 1, \dots, k$ に対し、 $e_j = (S_j, N_{j+1})$
- 各 $j = 1, \dots, k$ に対し、 $S_j \subseteq S \cup N_2 \cup \dots \cup N_{j-1}$

3. $t \in N_{k+1}$

これは順に,

1. パスに用いる各エッジは有向ハイパーグラフのエッジの集合の要素である.
2. パスに用いる各エッジの全ての入ノードは, すでに訪問済みのノードである.
3. パスの最後のエッジの出ノードに終点ノード t を含む.

ことを表す条件である.

また, 各 $j = 1, \dots, k$ に対し, 全ての e_j が単エッジの場合のパスを特に, 単パスと定義する.

2.2 有向ハイパーグラフ上での到達可能性判定問題

以下に, 本稿で扱う有向ハイパーグラフ上での到達可能性判定問題を定義する.

定義 1. 有向ハイパーグラフ上での到達可能性判定問題とは, 入力として始点ノード集合 S と終点ノード t が与えられたときに, S から t へのパスが存在するとき到達可能と出力し, それ以外は到達不可能と出力する問題である.

本稿では以降, 有向ハイパーグラフは非巡回かつ B-有向ハイパーグラフに限定する. 前者は, 有向ハイパーグラフ中の強連結成分を見つけ [Allamigeon 14], 一つのノードに置き換えることにより容易に非巡回グラフに変換することができるためであり, 後者は, 与えられた有向ハイパーグラフ DH の到達可能性判定と, DH が含む F-ハイパーエッジを単エッジに分割したグラフ DH' の到達可能性判定は同じ出力を返すことは問題の定義より自明であり, 一般性を失わないためである. また以降, 到達可能性判定問題と記述する際には, 有向ハイパーグラフ上での到達可能性判定問題のことを表す.

2.3 推移閉包情報

本稿で扱う推移閉包情報 (transitive closure information) とは, 有向ハイパーグラフ $DH(V, E)$ 上の考え得る全てのノード集合 $X \in \Pi(V)$ から, 有向ハイパーグラフ上の考え得る全てのノード $y \in V$ への到達可否を示す情報である. 考え得る始点ノード集合の数は $2^{|V|}$ 通り, 終点ノードの数は $|V|$ 通りあるため, 推移閉包情報は $O(|V| \cdot 2^{|V|})$ のメモリ使用量を要する. 一方, 前処理で推移閉包情報を保持しておくことにより, 入力が与えられた際には定数時間で到達可能性判定問題を解くことができる. また, 推移閉包情報は実際には行方向, 列方向が始点ノード集合, 終点ノードを表し, 行列の各要素を到達可能, 到達不可能を示す 1, 0 で表現する行列により実現され, これは推移閉包行列と呼ばれる.

簡潔推移閉包情報とは, 有向ハイパーグラフ $DH(V, E)$ 上のノード集合 $X' \in \text{tail}(E)$ から, 有向ハイパーグラフ上の考え得る全てのノード $y \in V$ の到達可否の情報である. 推移閉包情報では考え得る全ての始点ノード集合, つまりノードのべき集合 $\Pi(V)$ 全ての要素のノード集合からの到達可否の情報を保持していたが, 有向ハイパーグラフ (V, E) に対して, 考慮する始点ノード集合を $\text{tail}(E)$ に限定しているところが異なる.

単純推移閉包情報とは, 与えられた有向ハイパーグラフ $DH(V, E)$ から (B-) ハイパーエッジを取り除いた有向非巡回グラフに対して, 始点ノードとして考え得るノードから終点ノードとして考え得るノードへの, 全ての到達可能性情報と定義する. ここで, 単純推移閉包情報においては, 始点ノード集合ではなく, 始点ノードであることを注意する.

2.4 ZDD(ゼロサプレス型二分決定グラフ)

ここでは提案手法が用いる組合せ集合の圧縮表現である ZDD について述べる.

ZDD(ゼロサプレス型二分決定グラフ)[Minato 93] は, 有限集合の部分集合の集合を, 出次数が 2 である非循環有向グラフによって表現される. 別の表現をすると, ZDD は, グラフ構造による組合せ集合の表現とも言える. 組合せ集合とは, 「 n 個のアイテムから任意個を選ぶ組合せ」を要素とする集合である. ZDD は, 全てのアイテムについて場合分けした結果を二分決定木で表し, これを縮約することにより得られる.

ZDD の特徴として, 圧縮して組合せ集合の情報を持つためメモリ効率が良いことの他に, 二分木構造を圧縮して作られたグラフ構造のため, 任意の組合せがクエリとして与えられたときに, 含まれているか含まれていないかを, $O(n)$ で応答することができる点が挙げられる. また, ZDD の記憶量は表現する組合せ集合データの複雑さに依存し, 最悪の場合はアイテムの種類数に対して指数関数的なサイズとなるが, およそ人間が扱うような組合せ集合に対しては, 多くの場合, コンパクトに圧縮されたグラフになることが知られている. 組合せの性質にもよるが, 数十から数百種類ものノードから選んだ組合せを要素とする集合を, 汎用 PC のメモリ容量 (数 GB 程度) の範囲で現実的に表現することができる.

3. 提案手法

本節では, 2.3 節で指摘した推移閉包情報を保持する手法の問題点を改善し, クエリ応答と使用メモリ容量の適切なトレードオフを実現できるアルゴリズムを提案する. まず始めに 3.1 節で, ZDD を用いた推移閉包情報の圧縮アルゴリズムを提案する. 次に 3.2 節で, 全てのノードではなくある条件を満たす一部のノードに対して到達可能な始点ノード集合を保持することにより, 推移閉包情報のメモリ削減を行う方法を提案する. 最後に 3.3 節で, 保持する到達可能な始点ノード集合の情報を祖先ノードからなる組合せ集合に限定することにより, さらなるメモリ削減を実現する方法を提案する.

3.1 ZDD を用いた推移閉包情報の圧縮

本小節では, ZDD を用いた推移閉包情報の圧縮について提案する. 2.4 節では, ZDD は組み合わせ集合に特化した圧縮表現であることを述べた. ここで, 推移閉包情報は, 全てのノードに対し, 各ノードを終点ノードとした際の到達可能な始点ノード集合の情報の集まりであることに注目する. 以降, 任意のノードを終点ノードとした際の到達可能な始点ノード集合は, ノードの推移閉包情報と表す. 各ノードの推移閉包情報は, ノードの組合せ集合であるため ZDD で表現することができ, ZDD 表現されたノードの推移閉包情報を全てのノードに対して保持することで, ZDD 表現された推移閉包情報 TC_{ZDD} を得ることができる.

推移閉包情報を推移閉包行列で表し, 行列に対し連長圧縮や Repair などの 0,1 のビット列に対する圧縮を行う方法や, 推移閉包行列を疎行列表現で持つ方法と比べ, TC_{ZDD} では圧縮が行われつつも $O(|V|)$ 時間で行列の各要素へのアクセス, つまり任意のノード集合から任意のノードへの到達可否の情報を得ることができる点で優れている. より定式化すると, $\text{Reachable}(TC_{ZDD}, S, t)$ を, TC_{ZDD} を参照し始点ノード集合 S から終点ノード t への到達可否を真偽値で返す関数としたときに, Reachable 操作が $O(|V|)$ 時間で行えることを表す.

実行時には, クエリとして始点ノード集合 S と終点ノード t が与えられたときには, $\text{Reachable}(TC_{ZDD}, S, t)$ を実行する

だけで良い。

ZDD のサイズを $|ZDD|$ として表すと、到達可能性判定問題に対するクエリ応答は $O(|V|)$ 時間、メモリ使用量は $(|ZDD| \cdot |V|) = O(2^{|V|} \cdot |V|)$ 領域となる。ZDD のメモリ使用量は、前述の通り最悪の場合はノードの数に対して指数関数的なサイズになるが、多くの場合コンパクトに圧縮される。

3.2 ZDD を持つノードの選択

3.1 節では全てのノードに対して、各ノードの推移閉包情報を ZDD 表現し保持しておく手法を述べた。本小節では、ZDD 表現されたノードの推移閉包情報を、有向ハイパーグラフ $DH(V, E)$ に含まれる (F-) ハイパーエッジの集合 $E_f \subseteq E$ の出ノード集合 $head(E_f)$ に含まれるノードに限定した推移閉包情報 $TC_{ZDD.head}$ を保持することにより、さらなるメモリ使用量削減を目指す。また、実行時の別の補助構造として有向ハイパーグラフ $DH(V, E)$ に対する単純推移閉包情報 TC_{simp} を行列表現で保持しておく。

Algorithm1 に、本小節のアルゴリズムを示す。クエリとして始点ノード集合 S と終点ノード t が与えられたとき、まず始めに終点ノード t に対して、 $s_i \in S$ から t への単パスが存在するか TC_{simp} より、確認する。この時点で、もし存在すれば到達可能と出力する。次に、ハイパーエッジの出ノードの中で、終点ノード t へ単パスで到達可能な全てのノード t' に対して $Reachable(TC_{ZDD.head}, S, t')$ を行う。上記の全てのノード t' に対して、何れかのノードが、始点ノード集合 S より到達可能と判断されたときには、始点ノード集合 S から終点ノード t へ到達可能と出力し、それ以外は到達不可能と出力する。

メモリ使用量については、最悪時は $(|ZDD| \cdot |head(E_f)| + |V|^2) = O(2^{|V|} \cdot |V| + |V|^2)$ だが、疎な部分と密な部分が偏ったグラフにおいては $|head(E_f)| \ll |V|$ が多くの場合成り立ち、また web グラフなど実応用上のグラフはこのような特徴にあてはまるものが多いため実際的に効率の良い手法であるといえる。終点ノード t に到達可能なハイパーエッジの出ノードの数を $|H(t)|$ としたときに、クエリ応答は $O(|S| + |H(t)| \cdot |V|)$ 時間である。

3.3 祖先ノードの組合せ集合に限定した ZDD

まず始めに、有向ハイパーグラフ $DH(V, E)$ 中の任意のノード $x \in V$ に対する祖先ノード a を、以下のように定義する。

定義 2. 有向ハイパーグラフ $DH(V, E)$ 中の任意のノード $x \in V$ に対する祖先ノード a とは、 $DH(V, E)$ に含まれるハイパーエッジを単エッジに分解した有向非巡回グラフ $DAG(V, E')$ 上で、ノード x へ単パスで到達可能なノードである。

本小節で提案するアルゴリズムは、任意のノード x に対する推移閉包情報を、ノード x の祖先ノードのみを要素とする組合せ集合に限定する手法である。任意のノード x に対し x の祖先ノードのみを要素とした到達可能な全ての始点ノード集合を、ノードの祖先推移閉包情報と呼ぶ。また、ZDD 表現されたノードの祖先推移閉包情報を全てのノードに対して保持することで、ZDD 表現された祖先推移閉包情報 TC_ANC_{ZDD} を得ることができ、ZDD 表現されたノードの祖先推移閉包情報をハイパーエッジの出ノード集合に限定して保持した祖先推移閉包情報を $TC_ANC_{ZDD.head}$ で表す。

実行時の補助構造として、有向ハイパーグラフ $DH(V, E)$ に対する単純推移閉包情報 TC_{simp} と、全てのノードに対しての祖先ノード情報 ANC を保持する。 $v \in V$ に対し、 $ANC(v)$ はノード v の祖先ノードを集合を出力する。

Algorithm 1 ハイパーエッジの出ノードの推移閉包情報を用いたアルゴリズム

```

1: 入力: 始点ノード集合  $S$ , 終点ノード  $t$ , ハイパーエッジの出ノードの推移閉包情報  $TC_{ZDD.head}$ , 単純推移閉包情報  $TC_{simp}$ 
2: 出力: 到達可能, または到達不可能
3: procedure MAIN1( $S, t, TC_{ZDD.head}, TC_{simp}$ )
4:   for  $i = 1, \dots, |S|$  do
5:     if  $s_i \in S$  から  $t$  への単パスが存在 then  $\triangleright TC_{simp}$  参照
6:       出力: 到達可能
7:     end if
8:   end for
9:    $TC_{simp}$  を参照し,  $t$  に単パスで到達可能なハイパーエッジの出ノード全てを求める
10:  for  $t$  に単パスで到達可能なハイパーエッジの出ノード  $t'$  do
11:    if  $Reachable(TC_{ZDD.head}, S, t') = TRUE$  then
12:      出力: 到達可能
13:    else
14:      continue
15:    end if
16:  end for
17:  出力: 到達不可能
18: end procedure

```

Algorithm 2 ハイパーエッジの出ノードの祖先推移閉包情報を用いたアルゴリズム

```

1: 入力: 始点ノード集合  $S$ , 終点ノード  $t$ , ハイパーエッジの出ノードの祖先推移閉包情報  $TC\_ANC_{ZDD.head}$ , 単純推移閉包情報  $TC_{simp}$ , 祖先ノード情報  $ANC$ 
2: 出力: 到達可能, または到達不可能
3: procedure MAIN2( $S, t, TC\_ANC_{ZDD.head}, TC_{simp}, ANC$ )
4:    $S' = S \cap ANC(t) \triangleright$  始点ノード集合  $S$  のフィルタリング操作
5:   MAIN1( $S', t, TC\_ANC_{ZDD.head}, TC_{simp}$ ) を呼び出す
6: end procedure

```

Algorithm2 に本小節のアルゴリズムを示す。ハイパーエッジの出ノードの祖先推移閉包情報 $TC_ANC_{ZDD.head}$, 単純推移閉包情報 TC_{simp} , 祖先ノード情報 ANC とクエリとして始点ノード集合 S と終点ノード t が与えられたとき、初めに始点ノード集合に対するフィルタリング操作を行う。具体的には、始点ノード集合 S の要素の中から、終点ノード t の祖先ノードのみを抽出する。上記のフィルタリング操作で抽出した新たなノード集合を始点ノード集合とし、Algorithm1 を呼び出す。

到達可能性判定問題に対するクエリ応答は $O(|S| + |S| + |H(t)| \cdot |V|) = O(|S| + |H(t)| \cdot |V|)$ 時間、メモリ使用量は $O(2 + 2^2 + \dots + 2^{|V|} + |V|^2 + |V|^2) = O(2^{|V|+1} + |V|^2)$ である。

3.4 提案手法の比較

本小節では本節で提案した三つのアルゴリズムと、既存のアルゴリズムとの比較を行う。表 1 に、比較結果を示す。ただし、簡潔推移閉包情報 (STC) については、与えられるグラフとクエリの種類によって大きく依存し評価が難しいため、本小

表 1: 提案アルゴリズムの比較

	時間計算量	空間計算量
幅優先探索 (BF)	$O(E)$	$O(V + E)$
推移閉包情報 (TC)	$O(1)$	$O(V \cdot 2^{ V })$
簡潔推移閉包情報 (STC)	評価対象外	評価対象外
提案 1 (3.1 節)	$O(V)$	$(ZDD \cdot V) = O(2^{ V } \cdot V)$
提案 2 (3.2 節)	$O(S + H(t) \cdot V)$	$(ZDD \cdot head(E_f) + V ^2) = O(2^{ V } \cdot V + V ^2)$
提案 3 (3.3 節)	$O(S + H(t) \cdot V)$	$O(2^{ V +1} + V ^2)$

節の評価の対象からは外す。

クエリ応答に対する時間計算量に関しては、 $TC < \text{提案 1} \leq \text{提案 3} \leq \text{提案 2} < BF$ ，である。提案 2 と提案 3 に関しては最悪時間計算量は等しいが、提案 2 では各ノードの保持する ZDD の組合せ構成要素の数は常に $|V|$ 個であるが、提案 3 では各ノードに対して ZDD 表現された推移閉包情報は祖先ノードのみで構成されているため、ZDD の組合せ要素の数は祖先ノードの数であるため、ノードによってはより小さな ZDD になり、クエリ応答もその分早くなる。

空間計算量に関しては、 $BF < \text{提案 3} \leq \text{提案 2} \leq \text{提案 1} < TC$ ，である。ここで 2.4 節でも述べているが、ZDD のメモリ使用量は、最悪時はノードの数に対して指数関数的なサイズになるが、多くの場合コンパクトに圧縮される。また、提案 3 に関しては、 $|ZDD|$ を用いた評価をしていないが、これは各ノードによって祖先ノードの数が異なり、そのため ZDD のサイズもノードにより異なるからである。ただし、実際には任意のノード k の ZDD のサイズに関して $|ZDD_k| \leq |ZDD|$ が成り立つため、提案 3 は提案 2 よりメモリ使用量は優れており、最悪時空間計算量の観点からも改善していることがわかる。

4. 結論

本稿では、有向ハイパーグラフ上での到達可能性判定問題に対して、3 つのアルゴリズムを提案した。初めに、推移閉包情報を ZDD 表現したのち圧縮を行うことにより、メモリ使用量を減らしつつ、高速にクエリ応答を行うことができるアルゴリズムを議論した。次に、ハイパーエッジの出ノードを終点ノードの推移閉包情報に限定することにより、さらなる圧縮法を議論した。最後に、祖先ノードの概念を導入し考える組合せ集合を限定することにより、圧縮をさらに行う手法を議論した。

今後の課題として、提案のアルゴリズムを実データに対して計算機実験を行うことにより、詳細な解析を行いたい。具体的には、化学反応ネットワークや web グラフなどへの適用は興味深い課題である。

参考文献

[Allamigeon 14] Allamigeon, X.: On the Complexity of Strongly Connected Components in Directed Hypergraphs, *Algorithmica*, Vol. 69, No. 2, pp. 335–369 (2014)

[Ausiello 17] Ausiello, G. and Laura, L.: Directed hypergraphs: Introduction and fundamental algorithms - A survey, *Theor. Comput. Sci.*, Vol. 658, pp. 293–306 (2017)

[Minato 93] Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems, in *Proceedings of the 30th Design Automation Conference. Dallas, Texas, USA, June 14-18, 1993.*, pp. 272–277 (1993)

[Özturan 08] Özturan, C. C.: On finding hypercycles in chemical reaction networks, *Appl. Math. Lett.*, Vol. 21, No. 9, pp. 881–884 (2008)