CA-001

経路問題における頂点の不要性判定の計算困難性とアルゴリズム
Hardness and Algorithms for Irrelevant Vertex Detection in the s–t Path Problem

立花 真龍* 塩田 拓海† 斎藤 寿樹*
Shinryu Tachibana Takumi Shiota Toshiki Saitoh

1 はじめに

グラフとは，頂点集合 V と辺集合 E からなる組 (V, E) で表される離散構造であり，複雑な関係性をもつ実世界のデータを形式的に記述・解析するための基本的枠組みとして広く用いられている．その応用範囲は多岐にわたり，大規模な社会ネットワークにおける構造的性質の解析 [12]，ハイパーリンク構造に基づく検索エンジンの設計 [2]，ネットワーク構造における時間的変化を考慮したセキュリティ評価 [6] をはじめ，細胞内の分子間相互作用を表すシグナル伝達ネットワークの構造解析 [9, 10]，グラフ同型アルゴリズムに基づく化学構造の比較 [13]，サイクルや木といったグラフパターンに基づく特徴抽出を通じた，分子の活性などを判定する予測モデルの構築 [7] に至るまで多様である．

この中でも，2 頂点 s, t 間の単純パス（**s–t パス**）の数え上げは，古典的な問題であり，交通網やナビゲーションシステムにおけるルート提示 [3]，災害時における重要インフラの脆弱性評価 [11] など，実社会における多くの応用においても現れる．一方で，s–t パスの数え上げ問題は #P 完全であり，一般には指数時間を要する計算困難な問題である [14]．ゆえに，実社会で扱われる大規模グラフにおいては，前処理によって，s–t パスの数え上げに寄与しない，すなわち不要な頂点（および辺）を除去し，探索空間を削減することが極めて重要となる．

無向グラフにおいては s–t パスの数え上げにおける不要な頂点を除去する手法として，Block-cut tree (BC-tree) に基づくアプローチが知られている [1]．有向グラフにおいても，辺の向きを無視して無向グラフとして扱い，BC-tree に基づく手法を適用することで，一部の不要な頂点を削除できる場合がある．しかし，図 1 に示すような頂点は，s–t パスの数え上げに寄与しないにもかかわらず，BC-tree に基づく手法では削除できない．

問題設定．このような背景のもと，本研究では以下の判定問題を定式化する：

有向 s–t パスにおける頂点の不要性判定問題（Irrelevant Vertex in Directed Paths Problem：**IVDP**）
入力：有向グラフ $G = (V, E)$，および頂点 $s, v, t \in V$．
質問：頂点 v を通る s–t パスが存在しないか？すなわち，頂点 v は不要か？

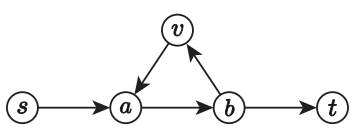

図 1 BC-tree に基づく前処理では除去できない頂点の例．この有向グラフにおいて，s–t パスは $s \to a \to b \to t$ の 1 通りだけであり，頂点 v は含まれない．

* 九州工業大学 Kyushu Institute of Technology
† 兵庫県立大学 University of Hyogo

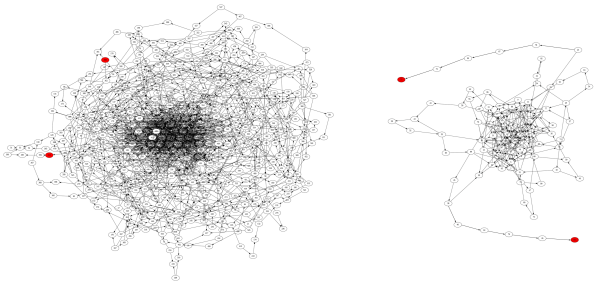

図 2 BC-tree に基づく前処理を行った有向グラフ（左図，557 頂点）と，左の有向グラフに対して設計したヒューリスティックを適用した結果（右図，90 頂点）．いずれのグラフも，赤色の頂点が端点を表す．

本研究の貢献．本研究では，IVDP が **coNP 完全** であることを示した．この結果から，IVDP は一般には多項式時間で不要な頂点を検出することは期待できないと考えられる．この困難性を踏まえ，ネットワークフローを用いた，任意の頂点 v に対して $O(|V| + |E|)$ 時間で不要性判定を行う**ヒューリスティック**を設計した．さらに，本手法を実装し，実世界の構造を模した複数の有向グラフに適用した結果，従来の BC-tree に基づく前処理では削除できなかった多数の頂点および辺を削除できることを確認した（図 2 参照）．

論文の構成．本稿は以下の構成で進める．まず，第 2 節では，基本的な用語の定義と記法を示す．続く第 3 節では，IVDP が coNP 完全であることの証明を示す．第 4 節では，IVDP に対するヒューリスティックを提案し，その有効性を計算機実験により評価する．最後に第 5 節で，結論と今後の課題を述べる．

2 基本的な用語と記法

2.1 グラフ

グラフ G は，頂点集合 V と辺集合 E により構成され，$G = (V, E)$ で表す．$E = \{(u, v) \mid u, v \in V, u \neq v\}$ が有向対からなるとき，G の辺は**有向辺**，G は**有向グラフ**という．**歩道**とは，頂点列 $\langle v_1, \ldots, v_k \rangle$ であり，$1 \leq i \leq k-1$ を満たす各 i について，$(v_i, v_{i+1}) \in E$ が成り立つものを指す．また，頂点に重複のない歩道を**パス**と呼ぶ．歩道またはパスにおいて，両端の頂点 v_1, v_k を**端点**と呼び，そのような歩道やパスを v_1–v_k 歩道，v_1–v_k パスと表記する．また，2 つのパス P_1, P_2 が互いに頂点を共有しないとき，これらは**点素**であるという．頂点 $v \in V$ が，v_1–v_k パスのいずれにも含まれないとき，v を**不要な頂点**と呼ぶ．2 頂点 $s, t \in V$ 間に s–t 歩道が存在すれば，s から t は**到達可能**といい，存在しなければ**到達不可能**という．有向グラフ $G = (V, E)$ に対し，3 頂点 $s, v, t \in V$ があり，s から t が到達可能であるとする．このとき，v を削除して s から t への到達可能性が失われる場合，v を s–t **カット点**という．グラフ $G' = (V', E')$ が $V' \subseteq V$ およ

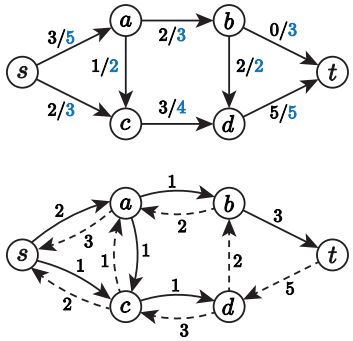

図 3　フローネットワーク (上図) と，その残余ネットワーク (下図)．フローネットワークにおいて，各辺におけるフローと容量を フロー/容量 の形で表す．残余ネットワークにおいて，実線は「追加で流せるフローの容量」を，点線は「既存のフローを逆方向に戻す量」を表す．

び $E' \subseteq E$ を満たすとき，G' を G の**部分グラフ**という．

2.2　ネットワークフロー

フローネットワークとは，有向グラフ $G = (V, E)$ に対し，各辺 $(u, v) \in E$ に非負の容量 $c(u, v) \in \mathbb{R}_{\geq 0}$ を割り当て，始点（**ソース**）$s \in V$ と，終点（**シンク**）$t \in V$ を指定した構造であり，(G, c, s, t) で表す．**フロー**とは，$f : E \to \mathbb{R}_{\geq 0}$ からなる関数であり，次の 2 つの条件を満たすものである．

(1). **（容量制約）** すべての辺 $(u, v) \in E$ に対して，$0 \leq f(u, v) \leq c(u, v)$.

(2). **（流量保存）** すべての $v \in V \setminus \{s, t\}$ に対して，v に流入するフローの総量と，v から流出するフローの総量が等しい：

$$\sum_{u \in V} f(u, v) = \sum_{w \in V} f(v, w).$$

フロー f において，ソース s からシンク t に送られるフローの合計を **フローの値**といい，次のように定義する：

$$|f| = \sum_{(s,v) \in E} f(s, v) - \sum_{(v,s) \in E} f(v, s).$$

最大フロー問題とは，与えられたフローネットワークにおいてフローの値を最大化するような f を求める問題である．この問題に対しては，Ford-Fulkerson 法 [8] や Edmonds-Karp アルゴリズム [4] などの手法が知られている．**残余ネットワーク**とは，あるフロー f に対して定義されるネットワークであり，各辺において「追加で流せるフローの容量」と「既存のフローを逆方向に戻す量」の両方を明示的に表現する．具体的には，元のネットワーク $G = (V, E)$ における各辺 $(u, v) \in E$ に対して，以下のような処理を行って構成される有向グラフ $G_f = (V, E_f)$ で表される（図 3 参照）：

- $f(u, v) < c(u, v)$ であるとき，容量 $c_f(u, v) = c(u, v) - f(u, v)$ を持つ辺 (u, v) を E_f に含める．
- $f(u, v) > 0$ であるとき，容量 $c_f(v, u) = f(u, v)$ を持つ辺 (v, u) を E_f に含める．

最大フロー問題と密接に関係する概念として，**s–t カット**がある．s–t カットとは，頂点集合 V を $S \subset V$ と

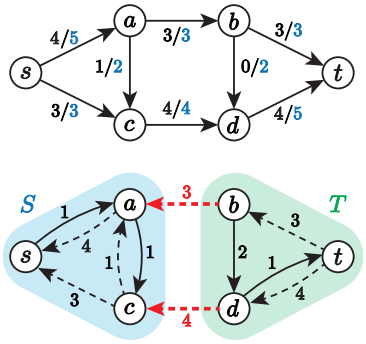

図 4　最大フローに対応するネットワーク（上図）と，その残余ネットワーク G_f および，G_f に基づく最小カット (S, T) (下図)．赤の点線はカット $E(S, T)$ に含まれる辺を表し，その容量は 7 で最大フローの値に一致する．

その補集合 $T = V \setminus S$ に分けることで定義され，$s \in S$, $t \in T$ を満たすものを指す．このとき，S から T に向かうすべての辺の容量の総和を **s–t カットの容量**と呼び，以下の式で与えられる（$E(S, T)$ は $u \in S, v \in T$ を満たすすべての辺 (u, v) の集合）：

$$c(S, T) = \sum_{(u,v) \in E(S,T)} c(u, v)$$

すべての s–t カット (S, T) の中で，カット容量 $c(S, T)$ が最小となるものを**最小カット**という．最大フローと最小カットの間には，次の関係が成り立つ：

定理 1（最大フロー最小カット定理 [8]）　任意のネットワークにおいて，ソース s からシンク t への最大フローの値は，最小カットの容量に等しい．

この定理に基づき，以下の手順で最小カットを求めることができる（図 4 参照）．

Step 1.　与えられたネットワーク上で，任意の最大フローを求める．

Step 2.　得られた最大フローに基づいて，対応する残余ネットワーク G_f を構築する．

Step 3.　残余ネットワーク G_f において，ソース s から容量が正の辺をたどって到達可能な頂点の集合 S を求める．

Step 4.　$T = V \setminus S$ として，元のグラフ G における s–t カット (S, T) を構成する．このときの $E(S, T)$ が，最小カットを与える辺集合となる．

3　IVDP の計算困難性

本節では，以下の定理を示す：

定理 2　IVDP は co-NP 完全である．

定理 2 の証明にあたり，まず，帰着元の問題である **有向グラフにおける 2 本の点素パス問題**を示す．その後，IVDP の補問題を定式化し，この補問題がクラス NP に属し，かつ NP 困難であることを示す多項式時間帰着を構成することで，IVDP が co-NP 完全であることを導く．

IVDP への多項式時間帰着を構成するために，以下の補助問題を導入する．

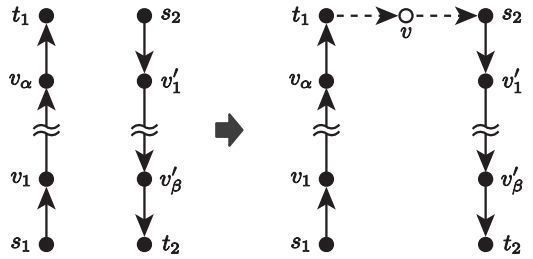

図 5　directed 2-DPP のインスタンス $G = (V, E)$（左図）を，VPDP のインスタンス $G' = (V', E')$（右図）に変換．白抜きの頂点が追加された頂点，点線の辺が追加された辺を表す．

有向グラフにおける 2 本の点素パス問題（2 Disjoint Paths Problem in Digraphs：**directed 2-DPP**） [5]

入力：有向グラフ $G = (V, E)$, および頂点 $s_1, t_1, s_2, t_2 \in V$.
質問：G において，s_1 から t_1 へのパス P_1 と，s_2 から t_2 へのパス P_2 が，互いに内部頂点を共有しない点素パスとして同時に存在するか？

この問題について，次の補題が知られている：

補題 3（[5] 補題 3）　directed 2-DPP は，NP 完全である．

続いて，IVDP の補問題を導入する．IVDP は「頂点 v を通る s-t パスが存在しないか」を問う否定の形の問題であり，その構造上，NP 完全問題から直接の帰着を構成することは難しい．そこで，本証明では「頂点 v を通る s-t パスが存在するか」を問う補問題を定式化し，この問題が NP 完全であることを示す．

有向 s–t パスにおける頂点の通過可能性判定問題（Vertex Passage in Directed Paths Problem：**VPDP**）

入力：有向グラフ $G = (V, E)$，および頂点 $s, v, t \in V$.
質問：頂点 v を通る s–t パスが存在するか？すなわち，頂点 v を通過して s から t に到達することが可能か？

以降，VPDP がクラス NP に属することの証明および，VPDP が NP 困難であることの証明を示す．

(I) VPDP がクラス NP に属することの証明

VPDP において答えが **Yes** である場合，その証拠として「頂点 v を通る s–t パス P」を与えることができる．ここで，確認すべき条件は次の通りである：

- 頂点列 P が，s から t への有向パスであること．
- 頂点 v がパス P の内部に含まれていること．

いずれの確認も，パス長は高々 $|V|$ であるため，多項式時間で実行可能である．ゆえに，VPDP はクラス NP に属する．

(II) VPDP が NP 困難であることの証明

VPDP の NP 困難性を示すために，directed 2-DPP からの多項式時間帰着を構成する．

(i) 帰着の構成

$(G = (V, E), s_1, t_1, s_2, t_2)$ を directed 2-DPP のインスタンスとする．ここで，$G = (V, E)$ に新たな頂点 v を追加し，辺 (t_1, v) および (v, s_2) を加える．つまり，$V' = V \cup \{v\}$, $E' = E \cup \{(t_1, v), (v, s_2)\}$ と定義し，$G' = (V', E')$ とする（図 5 参照）．この変換は，多項式時間で構成可能である．

(ii) 帰着の正当性

上述の構成によって得られる VPDP のインスタンスを元に，以下の二つが同値であることを示す．

(1) directed 2-DPP のインスタンス G において，s_1 から t_1 と s_2 から t_2 への点素な 2 つのパスが存在する

(2) VPDP のインスタンス G' において，頂点 v を通る s_1–t_2 パスが存在する

(1) ⇒ (2) の証明

$G = (V, E)$ において，s_1 から t_1 へのパスと，s_2 から t_2 へのパスが点素であると仮定する．このとき，これらのパスは，以下の形で記述できる：

- P_1: s_1 から t_1 へのパス
- P_2: s_2 から t_2 へのパス

P_1 と P_2 は互いに頂点を共有しないため，P_1 の終点 t_1 と P_2 の始点 s_2 を結ぶ追加の辺 (t_1, v) および (v, s_2) を用いて，s_1 から t_2 へのパス P を構成できる．ここで P は，P_1，頂点 v，および P_2 を順に接続したパスであり，v を通過している．ゆえに，$G' = (V', E')$ における VPDP の答えは **Yes** となる．

(2) ⇒ (1) の証明

$G' = (V', E')$ において，頂点 v を含む s_1–t_2 パス P が存在すると仮定する．このとき，P は，頂点 v に着目すると，以下の形に分解できる：

- P_1: s_1 から t_1 までの部分グラフ
- P_2: 辺 $(t_1, v), (v, s_2)$ からなるパス
- P_3: s_2 から t_2 までの部分グラフ

ここで，P_1 および P_3 は，グラフ G に含まれる頂点と辺のみから構成されるため，それぞれ s_1–t_1，s_2–t_2 のパスとなる．さらに，P がパスであることから，P_1 と P_3 は点素である．ゆえに，$G = (V, E)$ における directed 2-DPP の答えは **Yes** となる．

以上より，VPDP のインスタンス $G' = (V', E')$ において答えが Yes であることと，元の directed 2-DPP のインスタンス $G = (V, E)$ において答えが Yes であることは同値であることが示せた．また，この変換は多項式時間で構成可能であるため，VPDP は NP 困難と言える．

以上より，IVDP の補問題である VPDP の NP 完全であることから，IVDP は co-NP 完全である．

4　IVDP に対するヒューリスティック

3 節では，IVDP は co-NP 完全な問題であり，一般には多項式時間で不要な頂点を検出することは期待できないことを示した．本節では，IVDP に対して実用的な計算時間で動作し，実社会のグラフに対しても有効に機能するヒューリスティックを提案する．4.1 節では，提案手法の構成と手順を，擬似コードを用いて詳述する．4.2 節では，提案する手法が不要な頂点の検出に対して正しく機能すること（正当性）を示す．4.3 節では計算量の解析を行い，4.4 節では，提案手法の拡張と，検出が困難な構造の例を示す．最後に 4.5 節にて，計算機実験により提案手法の有効性を確認する．

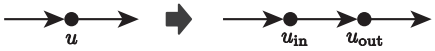

図 6　頂点 u（左図）を u_{in} と u_{out} に分割し，その間に容量 1 の辺 $(u_{\text{in}}, u_{\text{out}})$ を追加したグラフ（右図）．この変換により，各頂点にも容量制約を持たせたネットワークが構成できる．

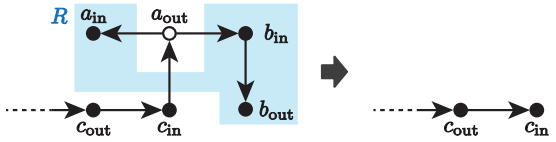

図 7　探索の起点 a_{out}（白抜きノード）から到達可能な頂点の集合 R（左図）と，左の残余ネットワークから $R \cup \{a_{\text{out}}\}$ を削除した後の構造（右図）．

4.1　ヒューリスティックの構成と実行手順

本節では，IVDP に対するヒューリスティックの構成と，その実行手順について説明する．以降では，アルゴリズム 1 に示した擬似コードに基づき，各手順の内容を詳しく説明する．

1 行目〜2 行目：初期化

不要な頂点の集合 U を s から到達不能な頂点で初期化し，V から U を削除する．

4 行目〜5 行目：頂点分割と容量の設定

まず，すべての辺に容量 1 を設定し，ネットワークを構成する．次に，各頂点 u を $u_{\text{in}}, u_{\text{out}}$ に分割する．その上で，$(u_{\text{in}}, u_{\text{out}})$ に容量 1 の辺を追加することで，頂点にも容量制約を持たせたネットワーク $(G', c, s_{\text{out}}, t_{\text{in}})$ を構築する（図 6 参照）．

6 行目〜11 行目：s から v への到達可能性の確認

ネットワーク G' 上で s_{out} から v_{in} への最大フローの一つを Ford-Fulkerson 法により計算する．最大フローが 0 の場合は，1 行目で到達不能な頂点として既に検出されているため考えなくてよい．また，最大フローが 2 以上の場合，本ヒューリスティックでは正当性を保証できないため，このケースは以降の議論から除外する．以後では，最大フローがちょうど 1 のとき，すなわち s–v パスが存在する場合に限定して考える．

その後，上記で求めた最大フローに対応する残余ネットワーク G_R を構築し，s_{out} から v_{in} に至るパスの頂点を，順に並べたものを π とする．

12 行目〜13 行目：s–v カット点の探索の初期化

s–v カット点とは，単一の頂点の削除によって 頂点 s_{out} から v_{in} への到達可能性が失われる頂点を表す（2.1 節参照）．カット点の探索に先立ち，カット点の集合 C を空集合で初期化し，探索の起点 u_{out} を s_{out}（π の先頭に位置する x_{out}）とする．

14 行目〜26 行目：s–v カット点の探索

最初に，15 行目〜20 行目から説明をする．ここでは，まず u を s–v カット点とし，集合 C に追加する（$u_{\text{out}} = s_{\text{out}}$ の場合は除く）．次に，残余ネットワーク G_R において，u_{out} から到達可能な頂点の集合 R を求める．その後，π から R および u_{out} を削除し，さらに，

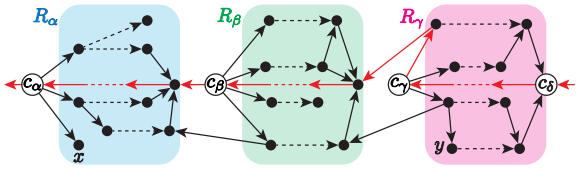

図 8　残余ネットワークにおいて，特定の最大フローに対して選択された頂点 c_i（$i \in \{\alpha, \beta, \gamma\}$）および，$c_i$ から到達可能な頂点の集合 R_i（$\alpha \to \beta \to \gamma$ の順で選択）．すべての辺の容量は 1 に設定されており，黒色の辺は追加でフローを流せる方向，赤色の辺は既存のフローを戻す方向を表す．

$R \cup \{u_{\text{out}}\}$ に含まれる頂点とそれらに接続する辺を，残余ネットワーク G_R から削除する（図 7 参照）．

21 行目以降では，次に探索の起点とする頂点 u_{out} を決定する．頂点列 π に x_{out} が存在する場合は，その先頭の x_{out} を探索の起点として処理を継続する．一方，x_{out} が存在しない場合は，探索を終了し，ループを抜ける．

27 行目〜30 行目：カット点削除後の到達可能性の確認

グラフ G から，カット点の集合 C に含まれる頂点とその接続する辺を削除し，部分グラフ G'' を得る．その後，G'' において v から t へのパスが存在するかを，深さ優先探索により確認する．パスが存在しない場合，v は不要な頂点であるとし，U に追加する．

4.2　正当性の証明

本節では，アルゴリズム 1 によって検出された頂点 v が，s–t パスの数え上げにおいて不要であることを示す．この主張を導くにあたり，まず，14 行目〜26 行目の s–v カット点の探索の手続きが，正しくカット点を検出していることを確認する．そこで，次の補題を導入する．

補題 4　ネットワーク $(G', c, s_{\text{out}}, v_{\text{in}})$ において，s_{out} と v_{in} の間の最大フローが 1 であるとき，ループ内で探索の起点として選択される頂点 u_{out}（s は除く）は，特定の最大フローに対する s–v カット点である．

証明　最大フローが 1 であることから，s_{out} と v_{in} の間のフローはパスとなっている．いま，u_{out} を起点としたとき，残余ネットワークにおいて到達可能な頂点の集合 R は，u–v パスの一部の流れを，R 内の頂点を経由する別のパスに流し直すことが可能であることを意味する（図 8 参照）．したがって，R に含まれる各頂点は，u–v パスおいて必ずしも経由する必要がないことを意味し，いずれも s–v カット点にはなり得ない．そこで，u_{out} および R の各頂点は，最大フローを構成する頂点列 π から除外される．

この操作の後，π に残る最初の頂点 y_{out} は，特定の最大フローを構成するパス上の頂点であるが，u_{out} からは残余ネットワーク上で到達不可能である．よって，y を通過しない限り s から v へのパスは存在せず，y は s–v カット点であることが示される．

加えて，y_{out} から v_{in} に至るパス上には，他にもカット点が存在する可能性がある．そこで，すでに探索済みである u_{out} および R を残余ネットワークから削除し，y_{out} を新たな起点として同様の操作を繰り返し行うことで，特定の最大フローに対するすべてのカット点が検出できる．　□

アルゴリズム 1 有向グラフにおける不要な頂点を検出するヒューリスティック

入力: 有向グラフ $G = (V, E)$，始点 s，終点 t
出力: 不要な頂点の集合 U

$U \leftarrow s$ から到達不能な頂点
$V \leftarrow V \setminus U$
for all $v \in V \setminus \{s, t\}$ **do**
    すべての辺 E に対し，容量を 1 に設定
    $G' \leftarrow$ 各頂点 u を $u_{\text{in}}, u_{\text{out}}$ に分割，有向辺 $(u_{\text{in}}, u_{\text{out}})$ を追加し，容量を 1 に設定
    $f \leftarrow G'$ 上で s_{out} をソース，v_{in} をシンクとして Ford-Fulkerson 法を実行
    if $f \geq 2$ **then**
        continue
    end if
    $G_R \leftarrow$ 最大フローに基づく残余ネットワーク
    $\pi \leftarrow$ 最大フローのパス上に現れる頂点の列（出現順）
    $C \leftarrow \emptyset$ ▷ s–v カット点の集合
    $u_{\text{out}} \leftarrow s_{\text{out}}$ ▷ 探索の起点
    while $u_{\text{out}} \neq \emptyset$ **do**
        if $u_{\text{out}} \neq s_{\text{out}}$ **then**
            $C \leftarrow C \cup \{u\}$
        end if
        $R \leftarrow$ 残余ネットワーク G_R 上で，u_{out} から到達可能な頂点の集合
        $\pi \leftarrow \pi \setminus (R \cup \{u_{\text{out}}\})$
        $G_R \leftarrow G_R$ から $(R \cup \{u_{\text{out}}\})$ およびそれに接続する辺を削除
        if π に x_{out} が存在する **then**
            $u_{\text{out}} \leftarrow \pi$ の先頭の x_{out}
        else
            $u_{\text{out}} \leftarrow \emptyset$
        end if
    end while
    $G'' \leftarrow G$ から C に含まれる頂点（および接続している辺）を削除したグラフ
    if v から t へのパスが G'' 上に存在しない **then**
        $U \leftarrow U \cup \{v\}$
    end if
end for
return U

次に，補題 4 によって得られたカット点の集合 C が，すべての容量 1 の最大フローに共通して現れること，すなわち s–v パス全体に対するカット点の集合となっていることを示す．この主張を形式化するため，以下の補題を導入する．

補題 5 補題 4 により得られたカット点の集合 $C = \{c_1, \ldots, c_k\}$ は，すべての容量 1 の s_{out} から v_{in} への最大フローが必ず通過する頂点の集合である．すなわち，C は s–v パス全体に対するカット点の集合である．

証明 背理法によりこれを証明する．仮定として，C が s–v パス全体に対するカット点となっていない，すなわち，C の頂点を通らないパスがあるとする．

まず，s–v パスが C の一部の頂点 c_i を通過しない場合を考える．簡単化のため，$c_0 = s_{\text{out}}$ とする．ここで，アルゴリズム 1 の 1 行目〜2 行目の初期化処理により，到達不能な頂点はあらかじめ削除されているため，すべての頂点はいずれかの R_i $(i \in \{0, 1, \ldots, k\})$ に含まれる．一方で，補題 4 の議論から，任意の $R_i \to R_j$（$i < j$）への有向辺は存在しない．すなわち，図 8 において，R_α 中の頂点 x から R_γ 中の y へ向かう有向辺は存在しない．よって，特定の頂点 c_i を回避するパスは存在せず，矛盾する．

次に，s–v パスが C のすべての頂点を通過しない場合を考える．これは，c_0 から到達可能な R_0 の頂点のみを通過して t_{in} まで到達可能なパスが存在している状況を表す．しかし，ここで得られるパスは，補題 4 により得られた特定の最大フロー（パス）と同時に実現でき，最大フローが 1 という条件に反する．ゆえに，このようなパスも存在しない． □

ゆえに，任意の s–v パスは，必ずカット点の集合 C を通過する必要がある．したがって，G から C を削除すれば，s から v への到達可能性は失われ，v を含むいかなる s–t パスも，v から t へのパスが存在しない限り成立しない．ゆえに，v から t へのパスが存在しなければ，v は s–t パスの数え上げにおいて不要な頂点と言える．

4.3 時間計算量の解析

本節では，アルゴリズム 1 における各ステップの実行時間を分析し，提案手法の時間計算量を導出する．

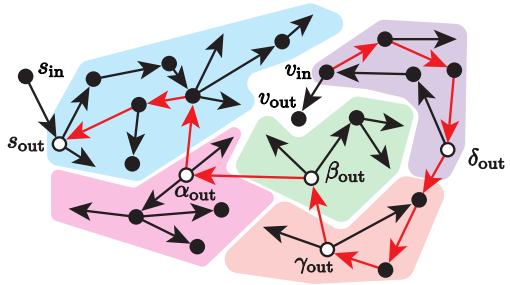

図 9　残余ネットワーク G_R に対する s_{out} からのカット点探索の流れの概略．赤色の辺は，既存のフローを戻す方向を表す．各反復において，u_{out} を起点として BFS により到達可能な頂点 R（色付き領域）を探索する．探索済みの成分は削除され，次の頂点 x_{out}（白抜き頂点）から処理が再開される（$\alpha \to \beta \to \gamma \to \delta$ の順）．この手順の繰り返しにおいて，すべての頂点・辺は高々一度しか探索されない．

1 行目〜2 行目：初期化

s から到達不能な頂点の探索は，幅優先探索（Breadth-First Search: BFS）により $O(|V|+|E|)$ で確認できる．

4 行目〜5 行目：頂点分割と容量の設定

各頂点 u を $u_{\text{in}}, u_{\text{out}}$ に分割し，対応する辺 $(u_{\text{in}}, u_{\text{out}})$ を追加する処理は $O(|V|)$ 時間で実行可能である．また，すべての辺の容量を 1 に設定する操作は $O(|E|)$ 時間で行えるため，このステップ全体の計算量は $O(|V|+|E|)$ となる．

6 行目〜11 行目：s から v への到達可能性の確認

G' 上で s_{out} から v_{in} への最大フローを，Ford-Fulkerson 法に基づくアルゴリズムで計算する．ただし本手法では，すべての辺と頂点の容量が 1 に設定されており，フローが 2 以上存在するかどうかを確認できれば十分である．そのため，増加路の探索を最大でも 2 回までに制限することで，計算量は $O(|V|+|E|)$ に抑えられる．

14 行目〜26 行目：s–v カット点の探索

この部分では，残余ネットワーク G_R 上で，u_{out} からの到達可能な部分グラフを用いて，s–v カット点の集合 C を段階的に抽出する．ここで，While ループ全体の計算量は $O(|V|+|E|)$ である．その理由は以下の通りである（図 9 参照）：

- 各ループでは，到達可能な頂点の集合 R を BFS によって計算するが，探索対象は G_R 上の未探索部分に限定される．
- 各頂点および辺は，G_R から削除される際に一度だけ処理されるため，BFS 全体の合計コストも高々 $O(|V|+|E|)$ に収まる．
- π の先頭の x_{out} を選択する操作は，キューなどの構造で保持することで $O(1)$ で行える．

このように，逐次的に分解された単一の探索であるため，計算量は $O(|V|+|E|)$ に抑えられる．

27 行目〜30 行目：カット点削除後の到達可能性の確認

カット点の集合 C に含まれる頂点とその接続する辺を元のグラフ G から削除し，部分グラフ G'' を構築する操作は，$O(|V|+|E|)$ 時間で実行可能である．その上で，G'' 上で v から t へのパスの有無を確認する処理は，BFS により $O(|V|+|E|)$ で確認できる．したがって，このステップ全体の計算量も $O(|V|+|E|)$ である．

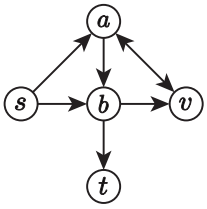

図 10　s–v パスに基づく手法では不要な頂点として検出できなかった頂点が，v–t パスに基づく手法により検出される例．s–v パスに対しては，カット点（単一の頂点の削除によって到達可能性が失われる頂点）の集合 C が空であり，削除が行えない．一方，v–t パスに対しては，カット点 a, b を削除することで，v が不要な頂点として検出される．

全体の計算量

初期化を除く，以上の手順を，すべての $v \in V \setminus \{s,t\}$ に対して繰り返すため，アルゴリズム全体の計算量は $O(|V| \cdot (|V|+|E|))$ となる．

4.4　ヒューリスティックの拡張と限界

本節では，これまでに述べたヒューリスティックに対して適用可能な拡張的な手法を紹介する．あわせて，本手法では検出できない不要な頂点の構造例を示し，その限界について考察する．

本ヒューリスティックは，ある頂点 v に対してまず s–v パスに着目し，その内部に現れるカット点の集合を取得する．そして，この集合を削除した後，v から t へのパスの有無を確認することで，v が不要な頂点であるかを判定するというものである．これに対して，v–t パスを先に用いてカット点の集合を構成し，その後に s–v パスの存在を調べるという，対称的な手法も考えられる．

このように判定の手順を切り替えることで，s–v パスに基づく手法では不要として検出できなかった頂点が，新たに検出される場合がある（図 10 参照）．そこで，s–v パスと v–t パスの両方に対して対称的に判定を行うことで，一方の手順では検出できなかった不要な頂点が，新たに検出される可能性がある．

さらに，不要な頂点として検出された頂点の集合 U に対して，それらを一括で削除するのではなく，逐次的に削除していくことで，グラフのサイズを段階的に縮小でき，後続の判定処理を高速化できる．このとき，削除の順序が後続の判定結果に影響しないことが重要である．以下の補題は，この性質を形式的に保証するものである．

補題 6　アルゴリズム 1 によって不要な頂点として検出された頂点の集合 U に対し，U の各要素を任意の順序で削除しても，最終的に得られる不要な頂点の集合は変わらない．

証明　任意の頂点 $v \in V \setminus \{s,t\}$ に対し，v が不要かどうかの判定は，s–v パスと v–t パスの存在に基づいて行われる．いま，v より先に，別の不要な頂点 v' が削除され

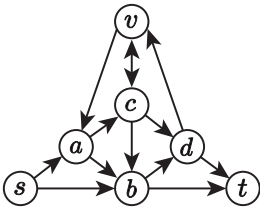

図 11 本ヒューリスティックでは削除できないグラフの例．s–v カット点および v–t カット点がいずれも存在しない．

たとする．このとき，v' を通過する s–t パスは，もともと存在しないことが不要性の判定によって保証されている．したがって，v' を削除しても，v を通過する s–t パスの存在には影響がない．一方で，v' を削除することで，残余ネットワークの構造が変化し，カット点の集合や BFS の経路に差異が生じる可能性はある．しかし，v の不要性の判定は，s–v および v–t の可到達性にのみ依存するため，最終的に v が不要かどうかの判定結果に違いは生じない．この議論は任意の $v, v' \in U$ に対して成り立つため，U 内の頂点は任意の順序で削除してよい． □

一方で，本ヒューリスティックでは削除できない構造をもつグラフも存在する．たとえば，図 11 に示す有向グラフでは，s–v カット点および v–t カット点の集合 C がいずれも存在しないため，本手法によって不要な頂点として検出することができない．

このような場合，複数の頂点からなるカット点の集合の導入が有効となる可能性があるが，複数の頂点からなるカット点の列挙には高い計算コストが伴う．したがって，複数の頂点を同時に削除しなければ到達可能性を遮断できないような構造に対しては，本手法では検出が困難となる．

4.5 計算機実験

本節では，提案したヒューリスティックを，実社会のデータに基づく有向グラフに適用し，その有効性を検証する．実験は，Intel Xeon CPU E5-2643 v4（3.40 GHz, 24 cores），メモリ 512 GB を搭載した Linux（CentOS 7.9）環境で実施した．評価対象には，グラフ数え上げコンペティション ICGCA 2024* にて提供された 50 個の有向グラフを用いた†．比較のため，辺の向きを無視して無向グラフとして扱い，BC-tree に基づく従来手法を適用した結果も併せて示す．計算機実験の結果の一部を 表 1 に示す‡．

この結果から，BC-tree に基づく従来手法よりも，多くの頂点を削除できることが分かる．とくに cit-HepPh-30-raw においては，従来手法により削除で削除できた割合が全体の約 0.5% にとどまるのに対し，提案手法では約 99.6% の頂点が削除可能であり，顕著な差が確認できる．また，図 2 に示すグラフは，提案手法によって効果的に削除が行えた p2p-Gnutella08-10-54-raw に対応している．

* International Competition on Graph Counting Algorithms 2024 の概要は https://afsa.jp/icgca2024/index.html を参照．
† ベンチマークは https://afsa.jp/icgca2024/results.html から入手可能である．
‡ 全ベンチマークに対する結果は https://afsa.jp/icgca2024/files/user02/user02.pdf を参照．

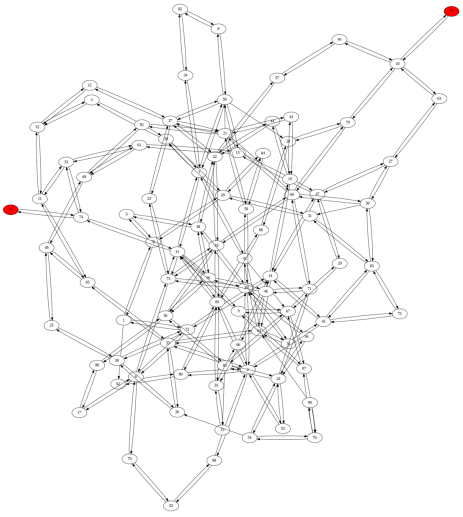

図 12 従来手法によって頂点が削除されたのち，提案手法ではいずれの頂点も削除できなかったグラフ．赤色の頂点は端点を示す．

一方で，BC-tree に基づく従来手法のほうが，多くの頂点を削除できる例も存在する．図 12 に示す rsndlib-36 は，従来手法の適用後に残ったグラフであり，このグラフに対しては，提案手法を適用しても，いずれの頂点も削除できなかった．この要因として，グラフ中に双方向の辺が密に存在し，構造的に無向グラフに近い形になっていることが挙げられる．このような場合には，提案手法よりも，従来手法のほうが有効に機能する傾向があると考えられる．

5 おわりに

本研究では，有向グラフにおける s–t パスの数え上げに寄与しない頂点の検出に着目し，Irrelevant Vertex in Directed Paths Problem（IVDP）を定式化した．IVDP は，特定の頂点 v を通る s–t パスが存在するか否かを判定する自然な問題であるが，その判定は coNP 完全であり，一般には多項式時間で不要な頂点を検出することは困難であることを示した．

この困難性をふまえ，IVDP に対するネットワークフローに基づくヒューリスティックを提案した．本提案手法は，時間計算量が $O(|V| \cdot (|V| + |E|))$ に抑えられ，大規模なグラフにも適用可能である．また，提案手法を実装し，実社会のデータに基づく有向グラフに適用した結果，従来の BC-tree に基づく手法では削除できなかった多数の頂点を効果的に検出できることを確認した．

今後の課題として，以下の二点が挙げられる．第一に，グラフ全体における「真に不要な頂点」がいくつ存在するかを求めることである．提案手法によって削除された頂点の数は把握できるものの，それが不要な頂点全体のどの程度に相当するかは分かっていない．ヒューリスティックの精度を評価するには，真に不要な頂点を厳密に特定することが必要である．第二に，入力グラフのクラスに着目することである．たとえば DAG やトーナメントグラフなどは，特有の構造的性質を持つ．これらのクラスのグラフに対しては，その性質を活かすことで，より強力かつ効率的なヒューリスティックの設計が可能になると考えられる．

表 1 BC-tree に基づく従来手法と，提案手法に基づく手法による削除される頂点の数およびその割合の比較（一部抜粋）

	入力グラフ		各手法適用後の頂点数と入力に対する割合			
ファイル名	頂点数	辺の数	従来手法	割合 [%]	提案手法	割合 [%]
airlines-migration-airrtraffic-airlines-8	235	2,101	192	81.7	190	80.9
cit-HepPh-10-30-raw	3,454	22,946	3,428	99.2	25	0.7
cit-HepPh-30-raw	10,363	139,133	10,316	99.5	37	0.4
email-EuAll-10-20-raw	26,521	120,582	10,763	40.6	6,084	22.9
email-EuAll-30-raw	79,564	220,304	24,342	30.6	12,061	15.2
matpower-case145-34	145	822	131	90.3	131	90.3
matpower-case1888rte-1420	1,888	4,500	784	41.5	779	41.3
matpower-case_ACTIVSg200-140	200	477	130	65.0	129	64.5
north-g.100.0	100	191	67	67.0	32	32.0
north-g.12.79	12	26	12	100.0	7	58.3
north-g.41.26	41	82	34	82.9	12	29.3
north-g.42.5	42	109	30	71.4	20	47.6
north-g.55.30	55	130	47	85.5	7	12.7
p2p-Gnutella04-10-76-raw	1,087	2,157	994	91.4	156	14.4
p2p-Gnutella04-30-raw	3,262	10,968	3,093	94.8	1613	49.4
p2p-Gnutella06-10-200-raw	871	1,952	795	91.3	86	9.9
p2p-Gnutella06-30-raw	2,615	8,707	2,455	93.9	1084	41.5
p2p-Gnutella08-10-54-raw	630	1,574	557	88.4	90	14.3
p2p-Gnutella08-30-raw	1,890	6,951	1,729	91.5	781	41.3
p2p-Gnutella24-10-118-raw	2,651	4,640	2,211	83.4	126	4.8
p2p-Gnutella24-30-raw	6,258	1,1759	5,154	82.4	469	7.5
p2p-Gnutella31-10-92-raw	18,775	54,012	15,326	81.6	7,023	37.4
rsndlib–36	100	300	90	90.0	90	90.0

謝辞

本研究は，JSPS 科研費 JP24970132, JP24973496, JP24KJ1816 の助成を受けたものです．

参考文献

[1] Etienne Birmelé, Rui Ferreira, Roberto Grossi, Andrea Marino, Nadia Pisanti, Romeo Rizzi, and Gustavo Sacomoto. Optimal listing of cycles and st-paths in undirected graphs. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1884–1896. SIAM, 2013. doi:10.1137/1.9781611973105.134.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. doi:10.1016/S0169-7552(98)00110-X.

[3] Matt Duckham and Lars Kulik. “simplest” paths: Automated route selection for navigation. In *Spatial Information Theory. Foundations of Geographic Information Science*, pages 169–185. Springer Berlin Heidelberg, 2003. doi:10.1007/978-3-540-39923-0_12.

[4] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972. doi:10.1145/321694.321699.

[5] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980. doi:10.1016/0304-3975(80)90009-2.

[6] Marcel Frigault, Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using dynamic bayesian network. In *Proceedings of the 4th ACM Workshop on Quality of Protection*, QoP ’08, pages 23–30, 2008. doi:10.1145/1456362.1456368.

[7] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’04, pages 158–167. Association for Computing Machinery, 2004. doi:10.1145/1014052.1014072.

[8] Lester R. Ford Jr. and Delbert R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

[9] Steffen Klamt, Julio Saez-Rodriguez, Jonathan A. Lindquist, Luca Simeoni, and Ernst D. Gilles. A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, 7:56, 2006. doi:10.1186/1471-2105-7-56.

[10] Steffen Klamt and Axel von Kamp. Computing paths and cycles in biological interaction graphs. *BMC Bioinformatics*, 10(1):181, 2009. doi:10.1186/1471-2105-10-181.

[11] Timothy C. Matisziw and Alan T. Murray. Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure. *Computers & Operations Research*, 36(1):16–26, 2009. doi:10.1016/j.cor.2007.09.004.

[12] Mark. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003. doi:10.1137/S003614450342480.

[13] Edward H. Sussenguth. A graph-theoretic algorithm for matching chemical structures. *Journal of Chemical Documentation*, 5(1):36–43, 1965. doi:10.1021/c160016a007.

[14] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979. doi:10.1137/0208032.