長さ l の s–t パス数え上げの XCC としての定式化

松本 吏司 [1]　原田 崇司 [1]　竹内 聖悟 [1]

[1] 高知工科大学 情報学群

高知工科大学
KOCHI UNIVERSITY OF TECHNOLOGY

2024 年 3 月 5 日

研究背景

s–t パス数え上げ問題

入力：無向グラフ $G = (V, E)$，頂点 $s, t \in V$
出力：s と t の間のパス数

- 部分グラフの列挙は組合せ論やグラフ理論における重要な問題
- パス数え上げは #P 完全

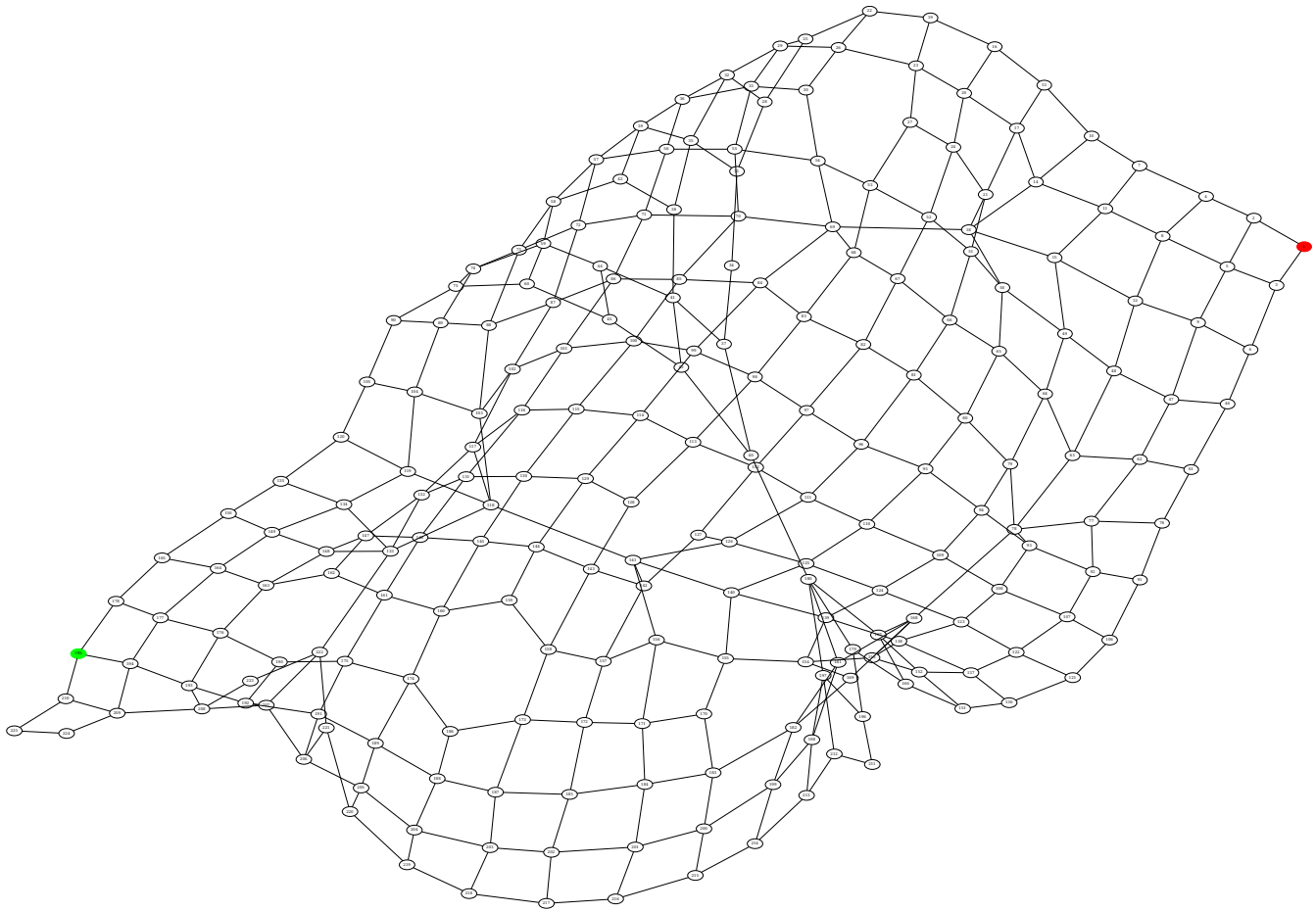
長さ l 以下の s–t パス数え上げ問題

入力：無向グラフ $G = (V, E)$，頂点 $s, t \in V$，正の整数 l
出力：s と t の間の長さ l 以下のパス数

研究背景

インスタンス例

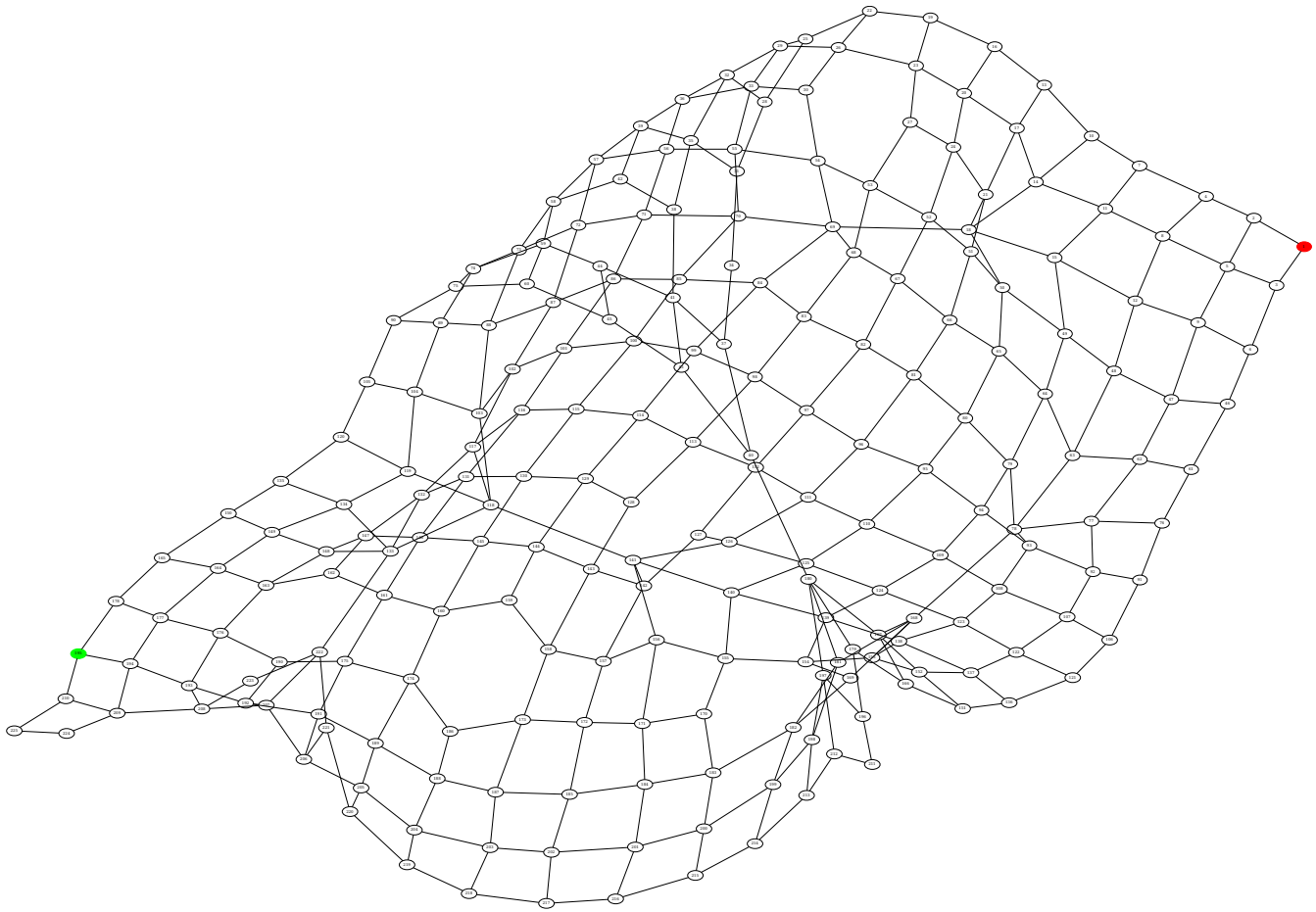
```
p edge 225 417
e 1 2
e 1 3
e 2 4
e 2 5
e 3 5
e 3 6
⋮
e 222 223
e 224 225
l 25
t 1 195
```



研究背景

インスタンス例

```
p edge 225 417
e 1 2
e 1 3
e 2 4
e 2 5
e 3 5
e 3 6
⋮
e 222 223
e 224 225
l 25
t 1 195
```



12147101 個

関連研究

s–t パス数え上げアルゴリズム

- バックトラッキング (Back Tracking)
  - DFS/BFS
- 動的計画法 (Dynamic Programming)
  - frontier-based search
- モデル計数 (Model Counting)
  - # SAT

ICGCA (2023 年に開催された国際グラフ数え上げ競技会)

長さ l 以下の s–t パス数え上げを行う

$n \times n$ の格子グラフにおける s–t パス数え上げ

ERATO 湊離散構造処理系プロジェクト により $n = 26$ (164 桁) まで計算[a]

[a] https://oeis.org/A007764/b007764.txt

研究背景

長さ l の s–t パス数え上げを色付き厳密被覆問題 (XCC) として定式化

- XC 及び XCC を高速に解くアルゴリズムを Knuth が提案[a]
- XC を解くアルゴリズムを発展させたものを Nishino らが提案[b]

[a] Donald E. Knuth, The Art of Computer Programming, Volume 4B:Combinatiorial Algorithms Part 2, Addison-Wesley Professional, 2022

[b] Nishino Masaaki, Yasuda Norihito, Nakamura Kengo, Compressing Exact Cover Problems with Zero-suppressed Binary Decision Diagrams, IJCAI, pp.1996–2004, 2021

XC (XCC) として上手く定式化できれば有望な手法

厳密被覆問題 (Exact Cover, XC)

厳密被覆問題

入力：集合 U と U の冪集合の部分集合を要素として持つ集合 S
出力：U の分割となる S の部分集合

集合 U の要素をアイテム，集合 S の要素をオプションと呼称
各アイテムを丁度一度被覆するオプションの組合せを求める問題

厳密被覆問題の例

$$U = \{a, b, c, d, e, f, g\}$$
$$S = \{\{c, e\}, \{a, d, g\}, \{b, c, f\}, \{a, d, f\}, \{b, g\}, \{d, e, g\}\}$$

$\{c, e\}, \{a, d, f\}, \{b, g\}$ は解
$\{c, e\}, \{a, d, g\}, \{b, c, f\}$ はアイテム c が重複するため解とならない
$\{a, d, g\}, \{b, c, f\}$ はアイテム e を被覆していないため解とならない

厳密被覆問題の拡張 (XC)

セカンダリアイテムの導入

Knuth により示された拡張で，アイテム集合を以下の二つに分割

- プライマリアイテム：丁度一回被覆しなければならないアイテム
- セカンダリアイテム：高々一回被覆してよいアイテム

拡張した厳密被覆問題の例

- プライマリアイテム：A, B, C, D
- セカンダリアイテム：x, y
- オプション：$\{A, x\}, \{A, B, y\}, \{A, C, x\}, \{B, D\}, \{C, y\}, \{D, x\}$

$\{A, x\}, \{B, D\}, \{C, y\}$ は解
$\{A, C, x\}, \{B, D\}$ は解
$\{A, B, y\}, \{C, y\}, \{D, x\}$ は y が重複するため解とならない

色付き厳密被覆問題 (Exact Cover with Colors, XCC)

色付き厳密被覆問題

XC を拡張した問題で，セカンダリアイテムに色を付与した問題
同じ色が付与されているアイテムは複数個取ってよい

各アイテムの制約

プライマリアイテム：ちょうど一度被覆しなければならない

セカンダリアイテム：色なしの場合，高々一度被覆してよい
色ありの場合，同じ色なら重複して被覆してよい

XCC は CSP を包含するクラス

色付き厳密被覆問題 (XCC)

色付き厳密被覆問題の例

セカンダリアイテム x に色 A を付与する場合，$x{:}A$ と示す

- プライマリアイテム：p, q, r
- セカンダリアイテム：x, y
- オプション：$\{p, q, x, y{:}A\}, \{p, r, x{:}A, y\}, \{p, x{:}B\}, \{q, x{:}A\}, \{r, y{:}B\}$

$\{p, r, x{:}A, y\}, \{q, x{:}A\}$ は解
$\{p, q, x, y{:}A\}, \{r, y{:}B\}$ は y の色が異なるため解とならない

XC, XCC を解くアルゴリズム

- Algorithm X (Exact cover via dancing links)
  - XC を解くためのバックトラックアルゴリズム
  - データ構造 Dancing Links を利用
  - 解の列挙を行う
- Algorithm C (Exact covering with colors)
  - Algorithm X を拡張し，XCC に対応したアルゴリズム
- Algorithm Z (Dancing with ZDDs)
  - Algorithm C に DP と ZDD を導入したアルゴリズム
  - 部分問題の表現にアイテムの選択状態を利用
  - 選択したアイテムの組合せを ZDD で表現
    - ZDD は 組合せ集合を簡潔に表現できるデータ構造

バックトラックアルゴリズムであるため，オプション数に影響を受ける

Dancing Links (XC, XCC を解くためのデータ構造)

バックトラックを行う際に，元の状態を復元する動作が手間
→ バックトラックを高速に行うデータ構造

- 双方向リンクの付け替えによるアイテムの削除・復元
- 削除したアイテムとその順番のみ保持

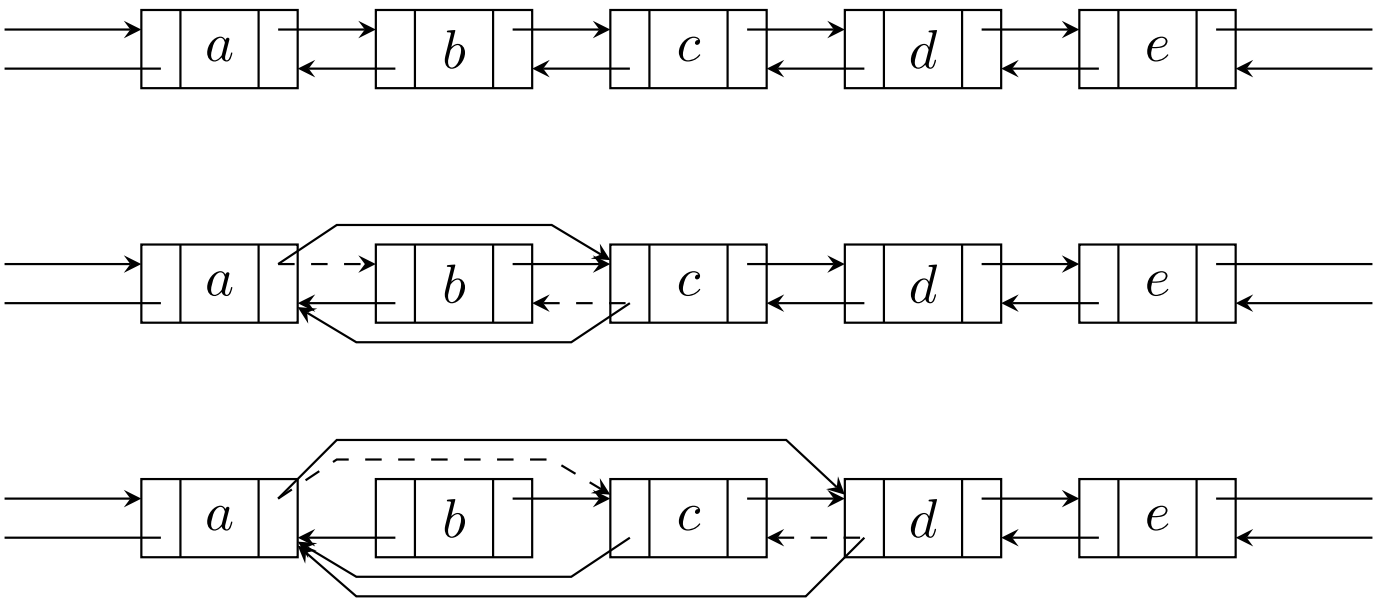


図 1: アイテム削除の例

提案手法

対象の問題

入力：無向グラフ $G = (V, E)$，頂点 $s, t \in V$，正の整数 l
出力：s と t の間の長さ l のパス数

XCC としての定式化

- プライマリアイテム：何番目に取る辺かを示す l 個のアイテム
- セカンダリアイテム：各頂点とパスの何番目の辺かを示す $|V| + l$ 個のアイテム
- オプション：パスの k 番目に辺 (v_i, v_j) を使うことを示すように生成

$$\{\#k,\ v_i{:}k-1,\ v_j{:}k,\ p_{k-1}{:}i,\ p_k{:}j\}$$

上記形式のオプションを $O(|E|l)$ 個生成

提案手法

$\{\#k, v_i : k-1,\ v_j : k,\ p_{k-1} : i,\ p_k : j\}$

$\{\#1,\ v_1 : 0,\ v_2 : 1,\ p_0 : 1,\ p_1 : 2\}$

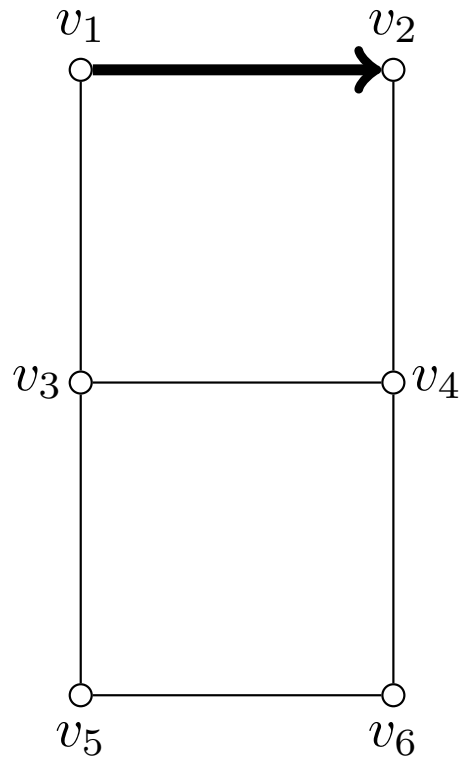
辺 (v_1, v_2) をパスの 1 番目として選択

$\{\#2,\ v_1 : 1,\ v_2 : 2,\ p_1 : 1,\ p_2 : 2\}$

辺 (v_1, v_2) をパスの 2 番目として選択

$\{\#3,\ v_1 : 2,\ v_2 : 3,\ p_2 : 1,\ p_3 : 2\}$

辺 (v_1, v_2) をパスの 3 番目として選択



$s = v_1,\ t = v_6,\ l = 3$

提案手法

$\{\#k, v_i : k-1,\ v_j : k,\ p_{k-1} : i,\ p_k : j\}$

$\{\#1,\ v_2 : 0,\ v_4 : 1,\ p_0 : 2,\ p_1 : 4\}$

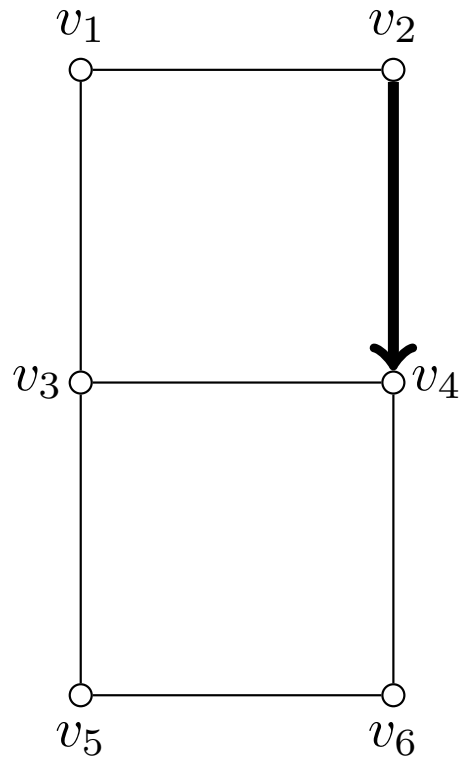
辺 (v_2, v_4) をパスの 1 番目として選択

$\{\#2,\ v_2 : 1,\ v_4 : 2,\ p_1 : 2,\ p_2 : 4\}$

辺 (v_2, v_4) をパスの 2 番目として選択

$\{\#3,\ v_2 : 2,\ v_4 : 3,\ p_2 : 2,\ p_3 : 4\}$

辺 (v_2, v_4) をパスの 3 番目として選択



$s = v_1,\ t = v_6,\ l = 3$

提案手法

$\{\#k, v_i : k-1,\ v_j : k,\ p_{k-1} : i,\ p_k : j\}$

$\{\#1,\ v_4 : 0,\ v_6 : 1,\ p_0 : 4,\ p_1 : 6\}$

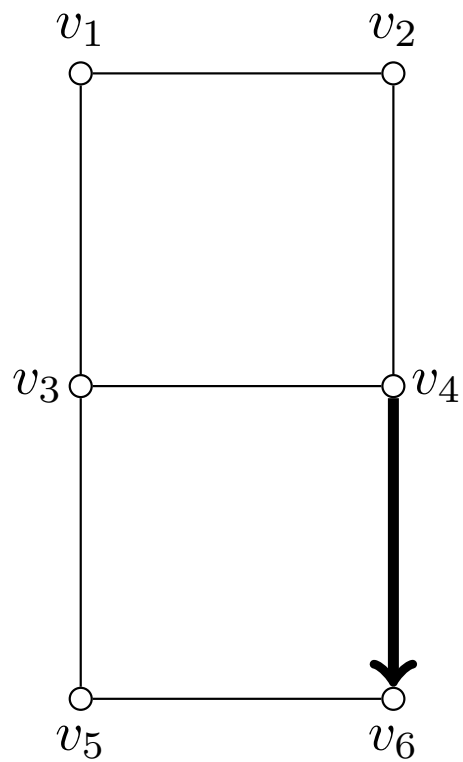
辺 (v_4, v_6) をパスの 1 番目として選択

$\{\#2,\ v_4 : 1,\ v_6 : 2,\ p_1 : 4,\ p_2 : 6\}$

辺 (v_4, v_6) をパスの 2 番目として選択

$\{\#3,\ v_4 : 2,\ v_6 : 3,\ p_2 : 4,\ p_3 : 6\}$

辺 (v_4, v_6) をパスの 3 番目として選択



$s = v_1,\ t = v_6,\ l = 3$

提案手法

~~$\{\#1,\ v_4:0,\ v_6:1,\ p_0:4,\ p_1:6\}$~~

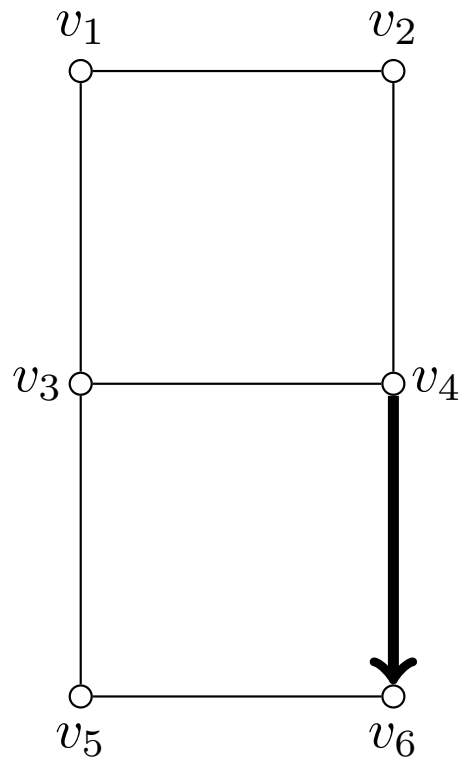
~~辺 (v_4, v_6) をパスの 1 番目として選択~~

~~$\{\#2,\ v_4:1,\ v_6:2,\ p_1:4,\ p_2:6\}$~~

~~辺 (v_4, v_6) をパスの 2 番目として選択~~

$\{\#3,\ v_4:2,\ v_6:3,\ p_2:4,\ p_3:6\}$

辺 (v_4, v_6) をパスの 3 番目として選択



$s = v_1,\ t = v_6,\ l = 3$

v_6 は端点 t であるためパスの 3 番目以外不要

提案手法

~~$\{\#1,\ v_2:0,\ v_4:1,\ p_0:2,\ p_1:4\}$~~

~~辺 (v_2, v_4) をパスの 1 番目として選択~~

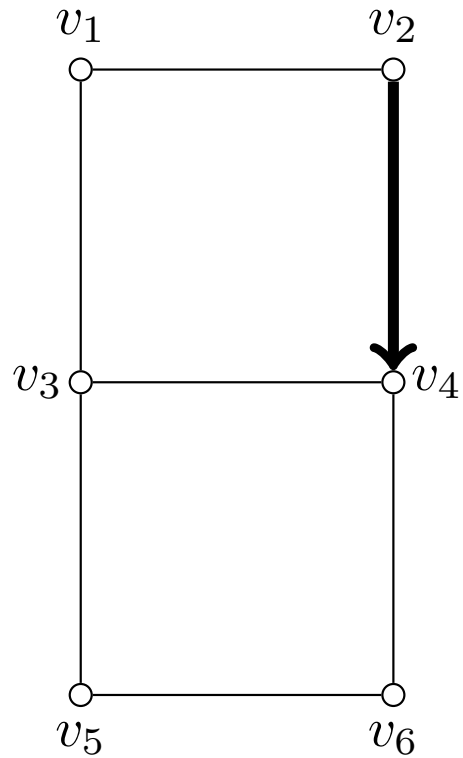
$\{\#2,\ v_2:1,\ v_4:2,\ p_1:2,\ p_2:4\}$

辺 (v_2, v_4) をパスの 2 番目として選択

~~$\{\#3,\ v_2:2,\ v_4:3,\ p_2:2,\ p_3:4\}$~~

~~辺 (v_2, v_4) をパスの 3 番目として選択~~

- v_1 から v_2 へは最低 1 つの辺が必要
  - パスの 1 番目として取れない
- v_4 から v_6 へは最低 1 つの辺が必要
  - パスの 3 番目として取れない



$s = v_1,\ t = v_6,\ l = 3$

提案手法

$\{\#1,\ v_1:0,\ v_2:1,\ p_0:1,\ p_1:2\}$

辺 (v_1, v_2) をパスの 1 番目として選択

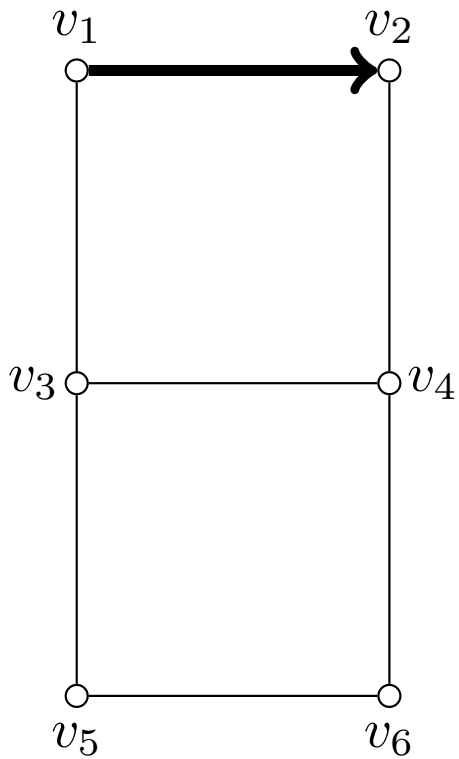
~~$\{\#2,\ v_1:1,\ v_2:2,\ p_1:1,\ p_2:2\}$~~

~~辺 (v_1, v_2) をパスの 2 番目として選択~~

~~$\{\#3,\ v_1:2,\ v_2:3,\ p_2:1,\ p_3:2\}$~~

~~辺 (v_1, v_2) をパスの 3 番目として選択~~

v_1 は端点 s であるためパスの 1 番目以外不要

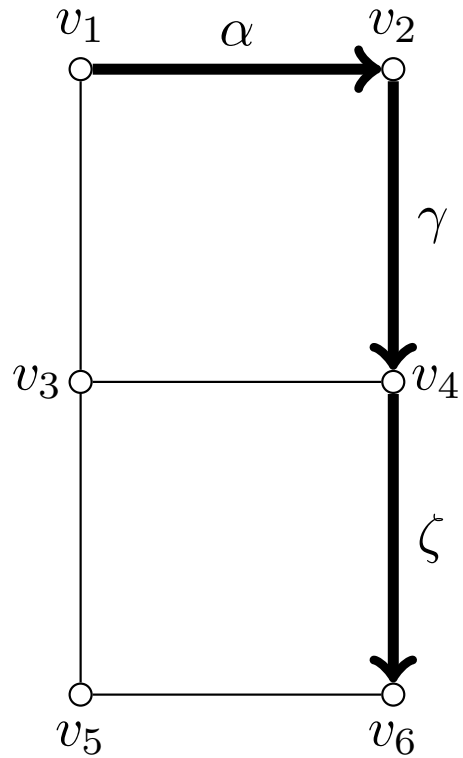


$s = v_1,\ t = v_6,\ l = 3$

提案手法

$$\{\#k, v_i : k-1, v_j : k, p_{k-1} : i, p_k : j\}$$

$$\alpha : \{\#1,\ v_1 : 0,\ v_2 : 1,\ p_0 : 1,\ p_1 : 2\}$$
$$\beta : \{\#1,\ v_1 : 0,\ v_3 : 1,\ p_0 : 1,\ p_1 : 3\}$$
$$\gamma : \{\#2,\ v_2 : 1,\ v_4 : 2,\ p_1 : 2,\ p_2 : 4\}$$
$$\delta : \{\#2,\ v_3 : 1,\ v_4 : 2,\ p_1 : 3,\ p_2 : 4\}$$
$$\varepsilon : \{\#2,\ v_3 : 1,\ v_5 : 2,\ p_1 : 3,\ p_2 : 5\}$$
$$\zeta : \{\#3,\ v_4 : 2,\ v_6 : 3,\ p_2 : 4,\ p_3 : 6\}$$
$$\eta : \{\#3,\ v_5 : 2,\ v_6 : 3,\ p_2 : 5,\ p_3 : 6\}$$



$$s = v_1,\ t = v_6,\ l = 3$$

オプション α, γ, ζ によりすべてのプライマリアイテムを被覆
(v_1, v_2, v_4, v_6) のパスに対応

提案手法

オプション数の削減

辺 (v_i, v_j) をパスの k 番目に取るとき

- s から v_i への最短経路長が l_s
  - $k < l_s$ である k 番目の辺として選択不可
- v_j から t への最短経路長が l_t
  - $l - l_t < k$ である k 番目の辺として選択不可

$l_s \leq k \leq (l - l_t)$ を満たす k についてオプションを生成

XCC の解の個数を求めるプログラム

Algorithm C を基本とし，列挙を行わずに数え上げを行う
Algorithm Z で導入された部分問題の共有
ZDD の構築は行わずに解の個数のみ記憶

実験環境

OS	CentOS Stream release 9
CPU	3.1GHz 8 コア Intel Core i9-9900
メモリ	64GB

ベンチマーク	インスタンス数
`one_pair`	100
`all_pairs`	50

ICGCA のベンチマーク[1] を用いて提案手法を評価
制限時間 10 分で長さ l 以下の s–t パス数え上げを行う
8 並列で各長さ k について実行 $(1 \leq k \leq l)$
比較に TdZdd[2] 及び ICGCA の上位 3 チームのプログラムを利用

[1] `https://afsa.jp/icgca/icgca-benchmarks.zip`
[2] `https://github.com/kunisura/TdZdd/tree/master/apps/ddpaths`

実験結果

ソルバ	アプローチ	one_pair	all_pairs
TLDC	BT, DP, MC	91	45
diodrm	BT, DP	85	47
TAG	DP	82	44
提案手法	**BT, DP**	**65**	**33**
TdZdd	DP	58	25

- 提案手法は合計 98 個を求解
- TdZdd よりも解けた個数は多い
  - 提案手法で解けたが TdZdd で解けないインスタンスが存在
  - 上記とは逆のインスタンスも存在
- コンテストの上位 3 チーム[3] には及ばない

[3] https://afsa.jp/icgca/results.html

提案手法の実験結果

解けなかったインスタンス

頂点数：45，辺数：187

長さ：45

長さ 45 のパスは存在しない

$l = 15$ まで求解

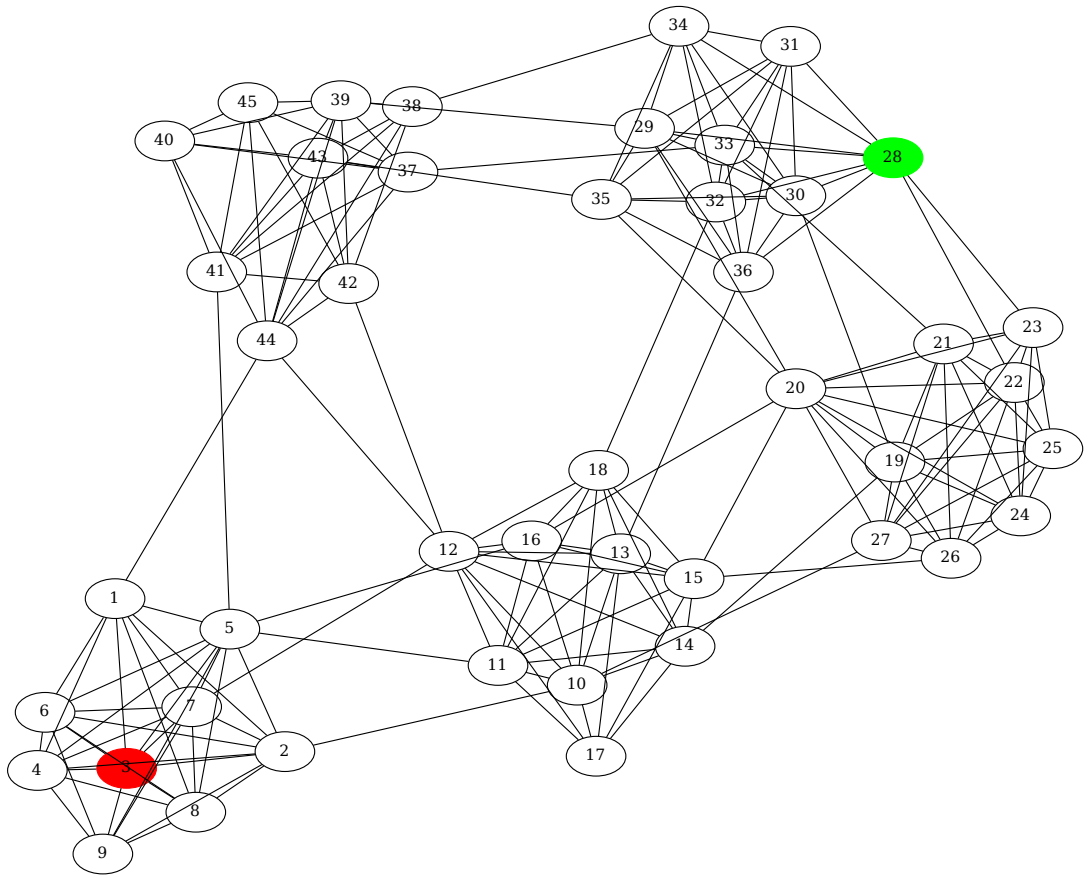


図 2: TdZdd では解けたグラフ

提案手法の実験結果

- 長さ l が大きいインスタンス
  - 色の候補が増える事によるシグネチャサイズの増大
  - 探索時にキャッシュする部分問題の増加
  - メモリ不足により CPU をフルに利用できない
- オプション数が 10000 から 15000 程度のインスタンス
  - 格子グラフのようなグラフ
  - $n = 17,\ 40 \leq l \leq 50$ までは求解可能
- オプション数が 3500 から 4000 程度のインスタンス
  - 複数のクラスタを形成するグラフ
  - 平均 $l = 15$ で解けない

うまくいかない定式化（非連結）

各頂点 v を高々 2 回取るように l 本の辺を被覆することだけを考えた定式化におけるオプション

$$\{\#k,\ v_i : k-1,\ v_j : k\}$$

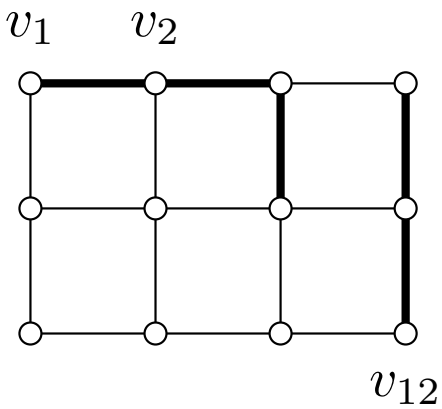


図 3: $l = 5$ の問題例

$\{\#1,\ v_1 : 0,\ v_2 : 1\}$

$\{\#2,\ v_2 : 1,\ v_3 : 2\}$

$\{\#3,\ v_3 : 2,\ v_7 : 3\}$

$\{\#4,\ v_4 : 3,\ v_8 : 4\}$

$\{\#5,\ v_8 : 4,\ v_{12} : 5\}$

$\{\#k,\ v_i : k-1,\ v_j : k,\ p_{k-1} : i,\ p_k : j\}$ のように，$p_{k-1} : i$ と $p_k : j$ が必要

うまくいかない定式化（閉路の発生）

- 各頂点 v_i がプライマリアイテム
- 生成されるパスにおける v_i の前後の頂点 p_i と n_i がセカンダリアイテム
  v_i を使わない場合は，p_i と n_i は無色

- 辺 (v_i, v_k) と (v_k, v_j) を使うことを意味するオプション
  頂点 v_k はパスに使われ，$(s, \ldots, v_i, v_k, v_j, \ldots, t)$ となることを意味

$$\{v_k,\ p_k{:}i,\ n_k{:}j,\ p_j{:}k\}$$

- 頂点 v_k を使わないことを意味するオプション

$$\{v_k,\ p_k,\ n_k\}$$

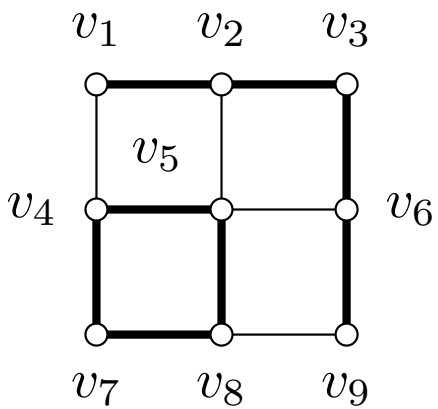


図 4: 閉路が発生

$\{v_1,\ n_1:2,\ p_2:1\}$

$\{v_2,\ p_2:1,\ n_2:3,\ p_3:2\}$

$\{v_3,\ p_3:2,\ n_3:6,\ p_6:3\}$

$\{v_6,\ p_6:3,\ n_6:9,\ p_9:6\}$

$\{v_9,\ p_9:6\}$

$\{v_4,\ p_4:7,\ n_4:5,\ p_5:4\}$

$\{v_5,\ p_5:4,\ n_5:8,\ p_8:5\}$

$\{v_8,\ p_8:5,\ n_8:7,\ p_7:8\}$

$\{v_7,\ p_7:8,\ n_7:4,\ p_4:7\}$

v_4, v_5, v_7, v_8 は $\{v_4, p_4, n_4\}, \ldots, \{v_8, p_8, n_8\}$ のオプションで被覆したい

まとめ

- 長さ l の s–t パス数え上げを XCC として定式化
- ICGCA のベンチマークを合計 98 個求解
- グラフの特性（パス幅など）は未使用

今後の課題

- 厳密被覆問題を解くアルゴリズムの高速化・拡張
  - 探索手法の並列化
  - D^3X へ適用するために XC としての定式化

    進展あり
  - D^3X の XCC への対応
  - SSXCC を用いた手法
- パス数え上げに関する課題
  - 長さ l 以下の数え上げの定式化

    進展あり
  - 被覆するアイテムの順序を最適化