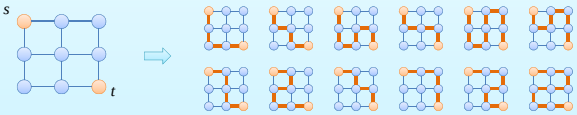
ZDD によるパスの列挙

川原 純[12]　斎藤 寿樹[12]　鈴木 拡[2]　湊 真一[21]　吉仲 亮[12]

[1]JST ERATO 湊離散構造処理系プロジェクト　[2]北海道大学情報科学研究科

パスの列挙



入力：グラフ $G=(V, E)$, 2頂点 s, t

出力：s から t までの**全て**のパス

応用

地理情報処理

ネットワークや論理回路の信頼性評価（故障検査）

ソフトウェアのフローチャートに沿ったテストケース作成

Fisher's Exact Test 等

既存研究

バックトラックを用いたアルゴリズム

パス1つあたり $O(|V| + |E|)$ 時間で出力 [Read and Tarjan, 75]

関連研究

パスの数え上げ

#P完全問題

Baxのアルゴリズム[Bax, 94]

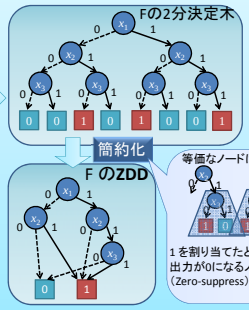
BDDを用いたパスのカウンティング [Sekine and Imai, 97]

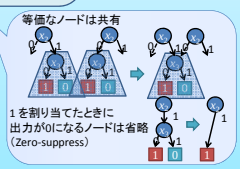
ZDD と集合演算

ZDD (Zero-suppressed binary Decision Diagram)
論理関数や組合せ集合を効率良く表現できるデータ構造

x_1	x_2	x_3	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

論理関数Fの真理値表





・ZDD を用いて、例えば{{a}, {b, c}, {a, c, d}} 等の組合せ集合（集合族）を効率良く保持可能

・さらに、集合の（代数）演算がZDDで効率良く実行可能（Family algebra）

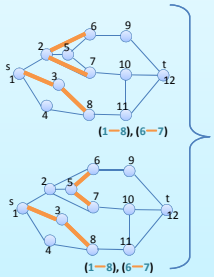
例えば、{{a}, {b, c}, {a, c, d}} に、各要素に e を加える演算を施すと {{a, e}, {b, c, e}, {a, c, d, e}} になる

・組合せ集合を多項式で表現するとこれらの演算が見やすい

(a + bc + acd) × e
= ae + bce + acde

・割算、剰余算も定義可能

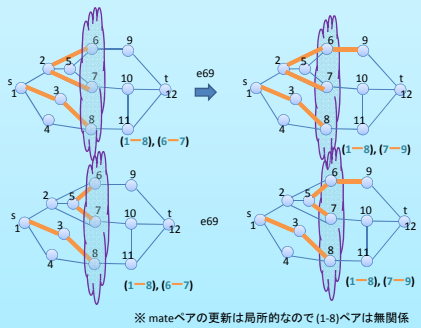
Mate-ZDD 法（提案手法）



パスマッチングの集合と mate（frontier のみ）を1つのZDD（組合せ集合）で表す

f = e13 × e38 × e26 × e27 × p18 × p67
+ e13 × e38 × e56 × e57 × p18 × p67

pij 変数で i と j が mate の関係であることを表す



※ mateペアの更新は局所的なので (1-8)ペアは無関係

mate の更新操作が ZDDの演算2〜3回で 全パスマッチングに対して一気に可能

f / p67

f の項のうち、p67 を含む項を取り出す

f / p67 × e69 × p79

アルゴリズム

初期値
f = p11 × p22 × p33 × … × p99

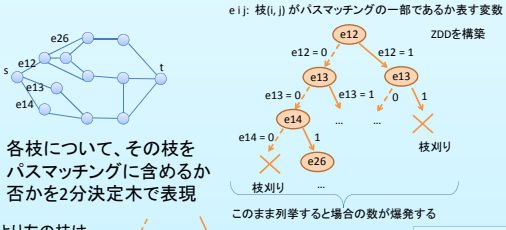
各枝 eij について
frontier を計算
frontier の中の各頂点 k,l について
f = f + f × eij / pik / pjl × pkl
頂点 i に関する処理を終えたとき，
頂点 i の次数が1ではいけない
f = f % pik
pii 変数の除去
f = f / pii + f % pii

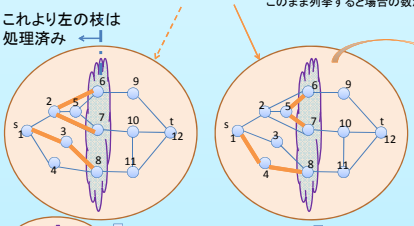
最後に
f = f / pst

%: 剰余を求める
例: (a b c + b c d + e f g) % c
= e f g

Knuth のアルゴリズム Simpath

The art of computer programming, vol. 4.1（Knuth著）掲載の ZDD を用いたパス列挙アルゴリズム

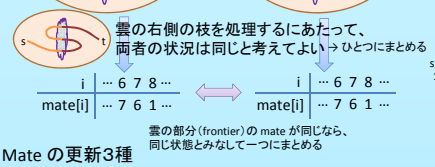


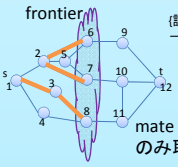


mate 配列

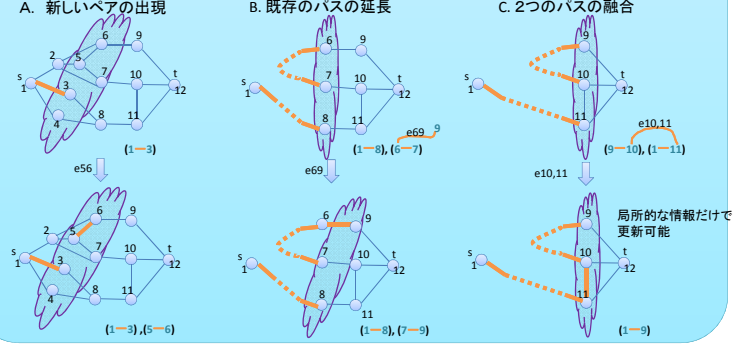
i	1	2	3	4	5	6	7	8	9	10	11	12
mate[i]	8	2	3	0	0	7	6	1	9	10	11	12

頂点がパスの端　→ 逆端の番号
頂点がパスの途中 → 0
（パスの無いところ → 長さ0のパスと考える）



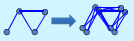


Mate の更新3種

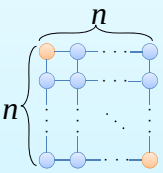


実験結果

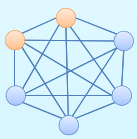
日本地図グラフ



2重化



格子グラフ



n 頂点完全グラフ

グラフ	Cypath	Simpath	Mate-ZDD	ノード数	パスの数
日本地図	>1000	<0.01	<0.01	850	5.1×10^{10}
日本地図2重化	>1000	24.01	248.62	1.7×10^{7}	5.5×10^{43}
ランダムグラフ（頂点数）：辺の生成確率=0.5 （100個生成した平均値）					
15	3.33	0.29	2.87	1.5×10^{5}	3.7×10^{6}
16	25.20	0.96	14.84	6.2×10^{5}	2.9×10^{7}
17	147.62	2.99	64.45	2.2×10^{6}	1.6×10^{8}
格子グラフ（n × n）					
8 × 8	>1000	0.03	0.44	31487	7.8×10^{11}
9 × 9	>1000	0.15	1.92	110215	3.2×10^{15}
10 × 10	>1000	0.63	6.00	377202	4.1×10^{19}
11 × 11	>1000	1.88	23.75	1.2×10^{6}	1.5×10^{24}
12 × 12	>1000	6.25	148.11	4.2×10^{6}	1.8×10^{29}
完全グラフ（頂点数）					
11	0.58	0.03	0.32	33830	9.9×10^{5}
12	5.79	0.07	1.34	121455	$9,9\times10^{6}$
13	64.30	0.42	8.51	435810	1.1×10^{8}
14	770.09	0.92	25.40	1.6×10^{6}	1.3×10^{9}
15	10049	2.74	110.07	5.6×10^{6}	1.7×10^{10}

計算時間（単位：秒）

Simpath、Mate-ZDD法、両者とも既存手法で列挙できない規模のグラフについてパスを列挙できる。Simpath は様々な種類のグラフに対して高速に列挙する。辺に制約条件をつけたグラフでは Mate-ZDD 法だと条件を記述しやすい。