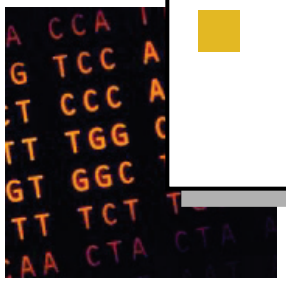
ERATO湊離散構造処理系プロジェクトの近況と今後の展望

湊　真一
北海道大学 情報科学研究科 / JST ERATO

2013年7月24日

本ERATOプロジェクトの基本構想



- 様々な工学的応用を持つ基盤技術として
  「離散構造処理系」に着目し、研究開発を行う

システム設計自動化　データマイニングと知識発見
大規模システム故障解析　機械学習と自動分類
制約充足問題　生命情報科学　web情報解析

工学的応用
→ 社会への影響大

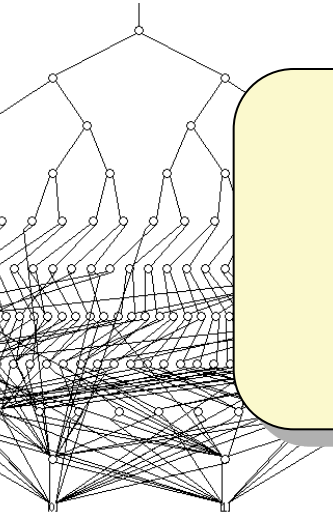
本研究の対象領域

離散構造処理系

集合理論　記号論理　帰納的証明
組合せ論　グラフ理論　確率論

性能向上
(10倍～100倍以上)

数学的概念構造



本ERATOプロジェクトの現況

- 2010年より研究活動を開始して以来、３年余が経過。
  - 残り2年たらず。今年度末に成果報告書提出
  - 開放的な組織で、多様な研究者が集まる場を形成
  - 従来の常識から外れた研究成果の出方
    （通常の評価尺度で測れない面白さ）

2009.10 2010.4 2010.10 2011.4 2011.10 2012.4 2012.10 2013.04

(0-th year) (1st year) (2nd year) (3rd year) (4th year)

採択
GL委嘱
交渉
研究員
スカウト
オフィス選定
内装工事
2013.07.24

キック
オフ
WS
FIT2010
企画
シンポ
秋の
合宿
情処
企画
シンポ
Knuth邸
訪問
各種の研究会
国際会議の共催
民間企業との共同研究
研究員出向交渉

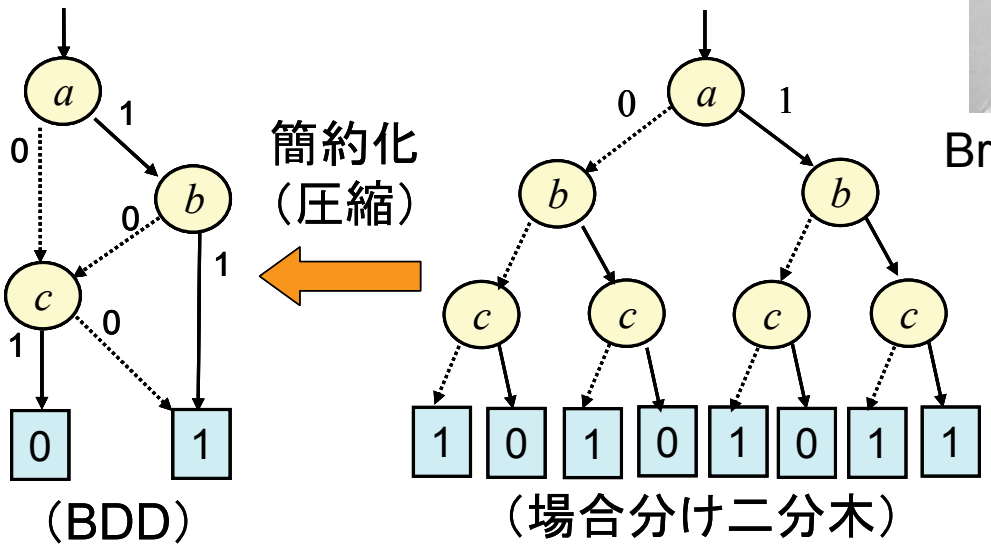
初夏の
WS
講究録
出版
秋の
合宿
通信
学会
企画
シンポ
各種研究会
共催
CMU
訪問
ALSIP
2011
電力網
報道発表

初夏の
WS
秋の
合宿
講究録
出版
公開
シンポ
AI学会
特集号
未来館
展示
ALSIP
2012
東工大
集中講義
北大連携講座

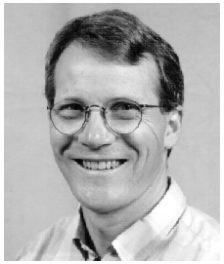
初夏の
WS
NII
講演
RM
2013

BDD（二分決定グラフ）

離散構造の最も基本的なモデルである「論理関数」の処理技法

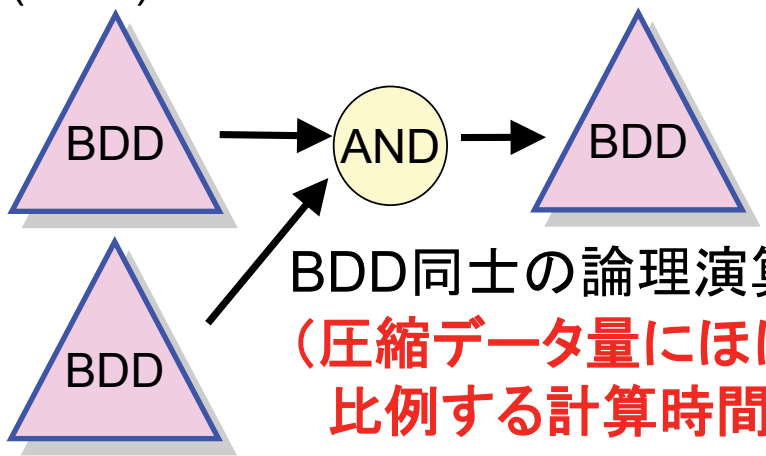


- 場合分け二分木グラフを簡約化（データ圧縮）
- **多くの実用的な論理データをコンパクトかつ一意に表現。**（数十～数百倍以上の圧縮率が得られる例も）



Bryant (CMU)

1986年に画期的なBDD演算**（Apply演算アルゴリズム）**を提案。
以後急速にBDD技術が発達。
（長期間、情報科学の全分野での最多引用文献となった）

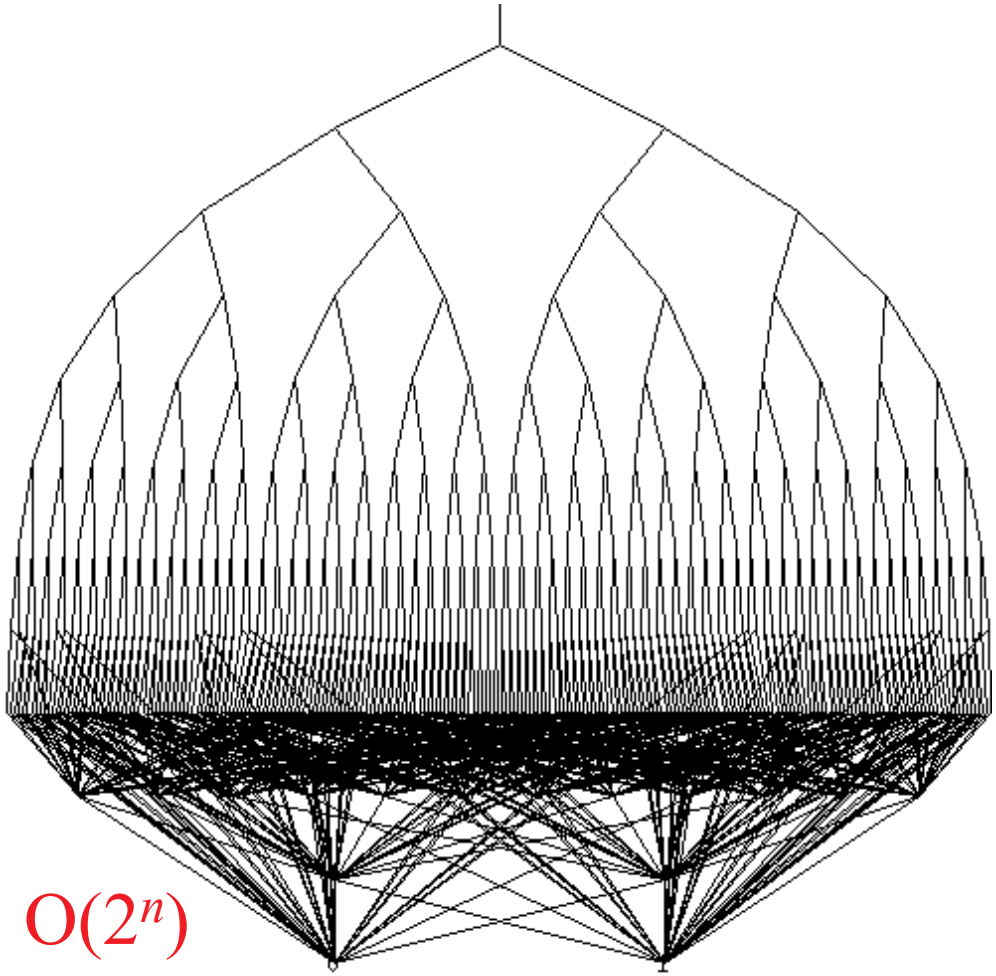


BDD同士の論理演算
（圧縮データ量にほぼ比例する計算時間）

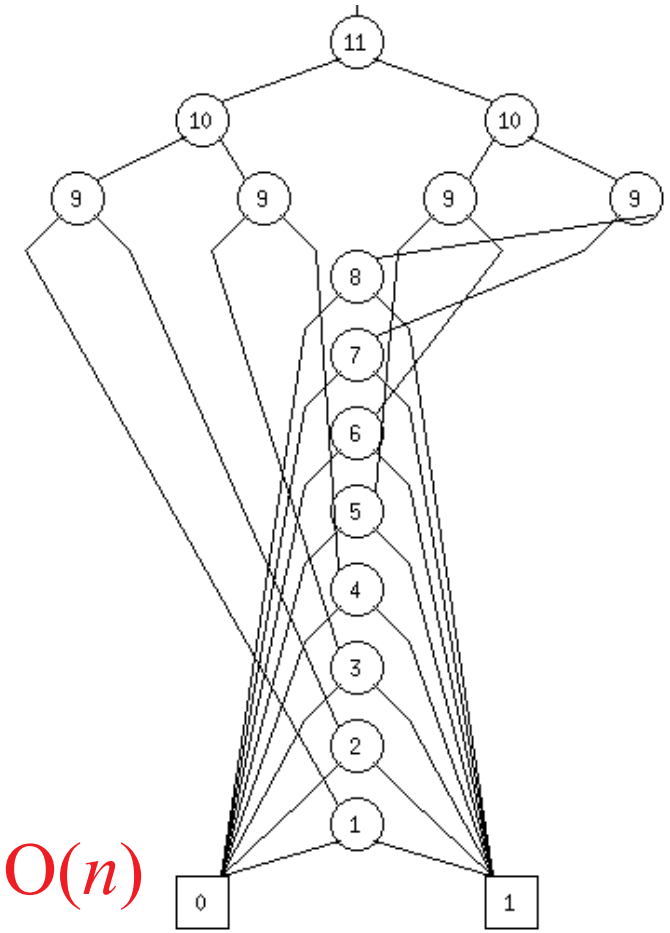
近年のPC主記憶の大規模化により、
BDDの適用範囲が拡大
（特に2000年以降）

BDD簡約化の効果

- 特定の問題では、指数関数的な圧縮効果が得られる。
  - 例題に依存するが、多くの実用的な問題で効果がある。



$O(2^n)$



$O(n)$

論理関数と組合せ集合

a	b	c	F	
0	0	0	0	
1	0	0	0	
0	1	0	0	
1	1	0	1	→ ab
0	0	1	1	→ c
1	0	1	1	→ ac
0	1	1	0	
1	1	1	0	

論理関数:

$$F = (a\ b\ {\sim}c)\ \mathsf{V}\ ({\sim}b\ c)$$

組合せ集合:

$$F = \{ab, ac, c\}$$

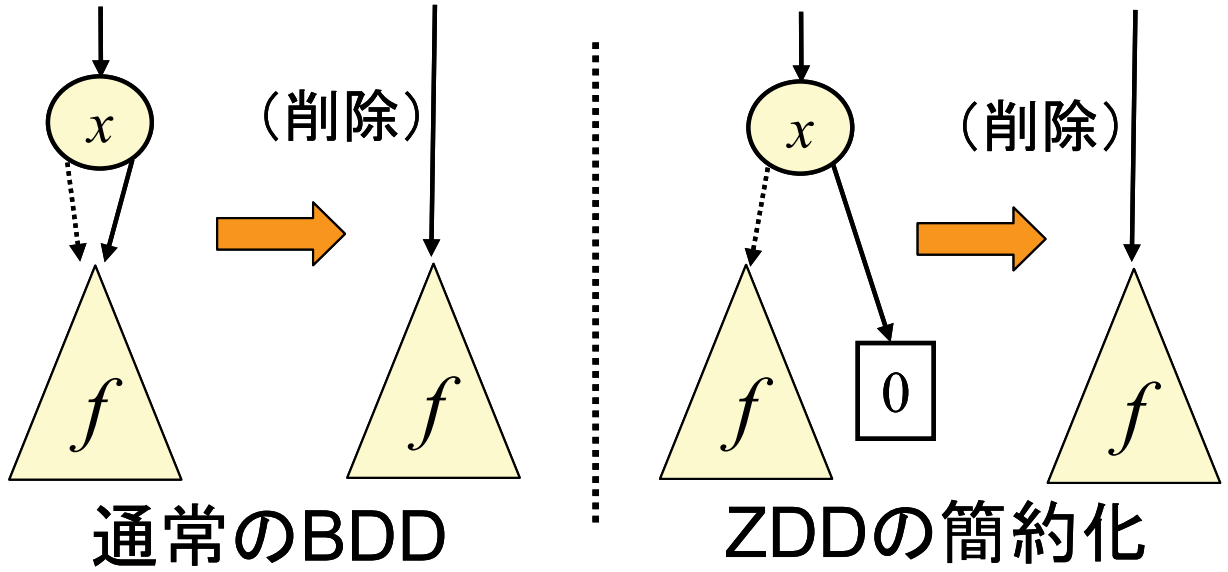
(買い物客の購入品)

- 組合せ集合と論理関数の演算は対応関係がある。
  - Union of sets → logical OR
  - Intersection of sets → logical AND
  - Complement set → logical NOT

ERATO

ZDD(ゼロサプレス型BDD)による集合の表現

- **「組合せ集合」**を効率的に表現するためのBDDの改良
  - **湊が世界で初めて考案し命名(1993年)**
  - **通常と異なる簡約化規則**を考案。
  - **疎な集合の族を扱う場合に著しい効果が得られる。**
    (例: 商店の陳列アイテム数に比べて1顧客の購入点数は極めて少ない。)



- ZDDはBDDの改良技術として現在、世界的に広く使われている。
  - 最近では、データマイニング分野に応用されて、画期的な有効性が示されている。(数百倍のデータ圧縮率・数十倍の処理高速化)
  - 他にも応用例は増えつつある。

Comparison of BDDs and ZDDs

- Many of real-life problems are likely **asymmetric**.

(Asymmetric world)

- Data mining & Knowledge discovery
- Market data analysis
- Web data analysis
- Formal concept analysis
- Risk analysis
- Calculation with Bayesian networks
- Machine learning & clustering
- etc.

(Symmetric world)

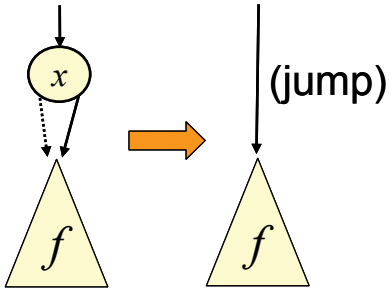
- VLSI Logic Design
- Formal Verification
- etc.

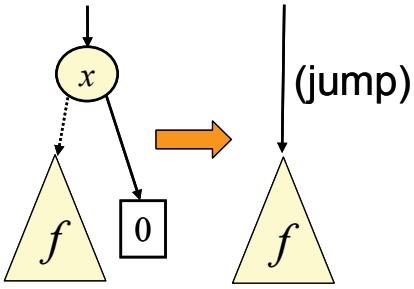
ZDD

BDD

Symmetric reduction

Asymmetric reduction

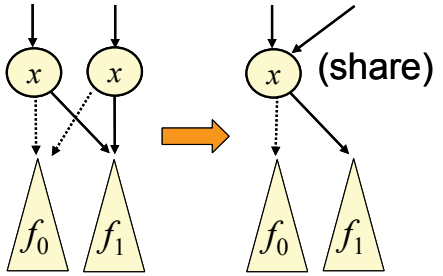




Quasi-reduced BDD

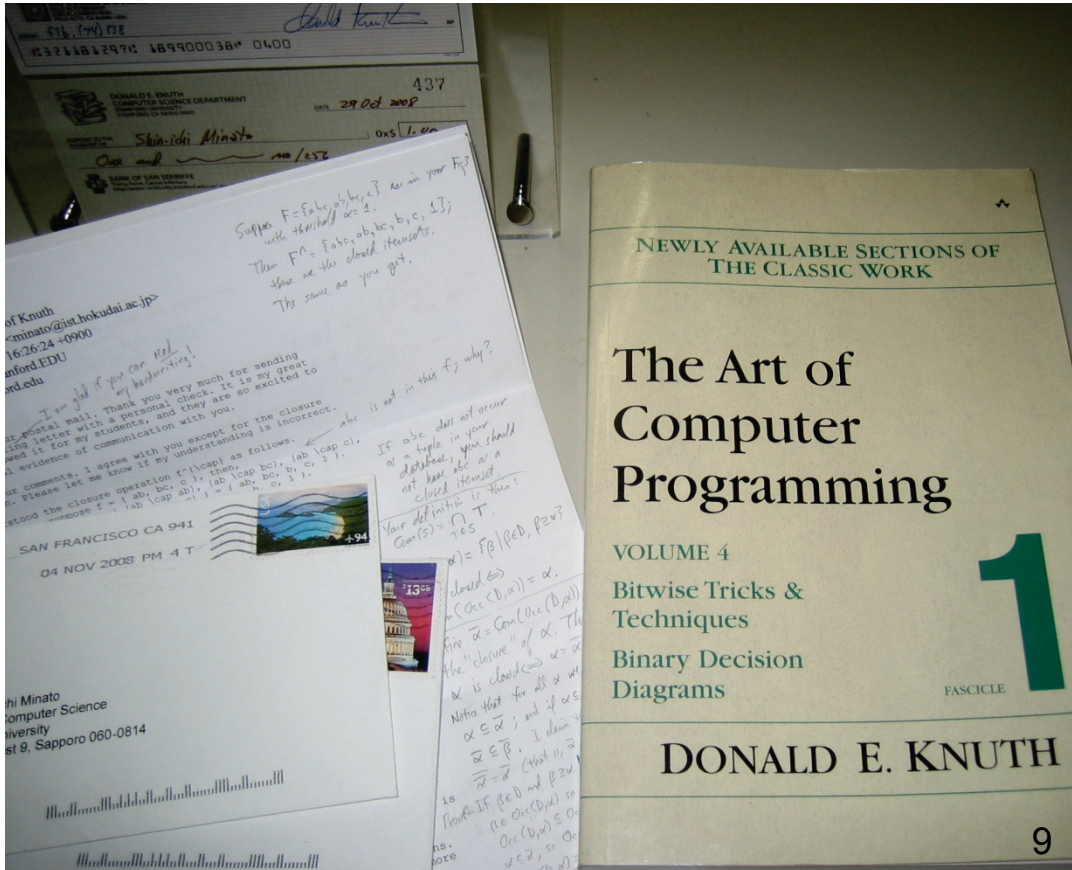
Sub-graph sharing (Dictionary-based)

Naïve representation



Knuthの名著とBDD/ZDD

- Knuthの世界的名著「The Art of Computer Programming」の最新巻（Vol.4, Fascicle 1, 2009）で、BDDが取り上げられ、その中でZDDが30ページ以上に渡り詳しく解説された。
- **日本人の研究成果が、このシリーズに項目として詳細に掲載されるのは初めて。**
- Knuth氏本人から、ZDD考案者として校正作業への協力を依頼する長文のメールと手紙を受領。
- 2010年5月には湊がKnuth邸を訪問し、プロジェクトの方向性について意見交換を行った。



ERATO

Algebraic operations for ZDDs

- Knuth evaluated not only the data structure of ZDDs, but more interested in **the algebra on ZDDs**.

φ, $\{1\}$	***Empty*** and ***singleton set***. (0/1-terminal)
$P.top$	Returns the ***item-ID*** at the top node of P.
$P.onset(v)$ $P.offset(v)$	Selects the subset of itemsets including or excluding v.
$P.change(v)$	Switching v (***add / delete***) on each itemset.
$\cup$, $\cap$, $\setminus$	Returns ***union, intersection,*** and ***difference set***.
$P.count$	***Counts number*** of combinations in P.
$P * Q$	***Cartesian product set*** of P and Q.
P / Q	***Quotient set*** of P divided by Q.
$P \% Q$	***Remainder set*** of P divided by Q.

Basic operations (Corresponds to Boolean algebra)

New operations introduced by Minato.

Formerly I called this "unate cube set algebra," but Knuth reorganized as **"Family algebra."**

Useful for many practical applications.

ZDDの応用

- 元々はLSI設計の論理式(CNF/DNF)の簡単化に利用
  - 膨大な項数の論理式を高速に因数分解する方法[Minato96]
  - 算術式の因数分解法[Minato97]
- データマイニングへの応用
  - ZDDを用いた頻出パタンマイニング[Minato2005]
  - LCM over ZDDs アルゴリズム[Minato-Uno2008]
  - 時分割データベースからのパタン変化の検出[Minato2010]
- グラフに関する種々の問題への応用
  - 最大クリーク問題、彩色問題、カバー問題等[Coudert97]
  - ZDDを用いた超高速パス列挙アルゴリズム[Knuth2009]
  - 電力ネットワークの制御、避難所配置問題、通信NW、他

ZDD応用例：頻出アイテム集合抽出問題

- データマイニングの最も基本的な問題
  - 最小出現頻度α以上のレコードに含まれるアイテム組合せの部分集合を抽出・列挙する問題

レコード番号	アイテム集合
1	a b c
2	a b
3	a b c
4	b c
5	a b
6	a b c
7	c
8	a b c
9	a b c
10	a b
11	b c

最小頻度 α = 10 → { b }

最小頻度 α = 8 → { ab, a, b, c }

最小頻度 α = 7 → { ab, bc, a, b, c }

最小頻度 α = 5 → { abc, ab, bc, ac, a, b, c }

最小頻度 α = 1 → { abc, ab, bc, ac, a, b, c }

“LCM over ZDDs” [Minato et al. 2008]

- ***LCM:*** *[Uno2003]*
  Output-linear time algorithm of frequent itemset mining.
- ***ZDD:*** *[Minato93]*
  A compact graph-based representation for large-scale sets of combinations.

Combination of the two techniques

Generates large-scale frequent itemsets on the main memory, with a very small overhead from the original LCM.

(→ Sub-linear time and space to the number of solutions when ZDD compression works well.)

LCM-ZDD 法による高速化

- 計算結果の頻出アイテム集合を、メモリ上に圧縮してZDDで表現し、そのポインタのみを返す。
  - 計算結果をファイルに出力しない。

レコード番号	アイテム集合
1	$a\ b\ c$
2	$a\ b$
3	$a\ b\ c$
4	$b\ c$
5	$a\ b$
6	$a\ b\ c$
7	c
8	$a\ b\ c$
9	$a\ b\ c$
10	$a\ b$
11	$b\ c$

LCM-ZDD法

(最小頻度 $\alpha = 7$)

$\{ ab, bc, a, b, c \}$

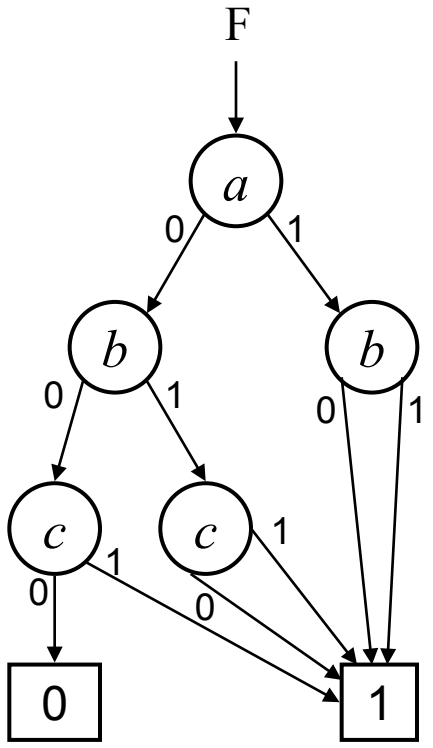


Table 2. Co…CM ov…ith the prev…

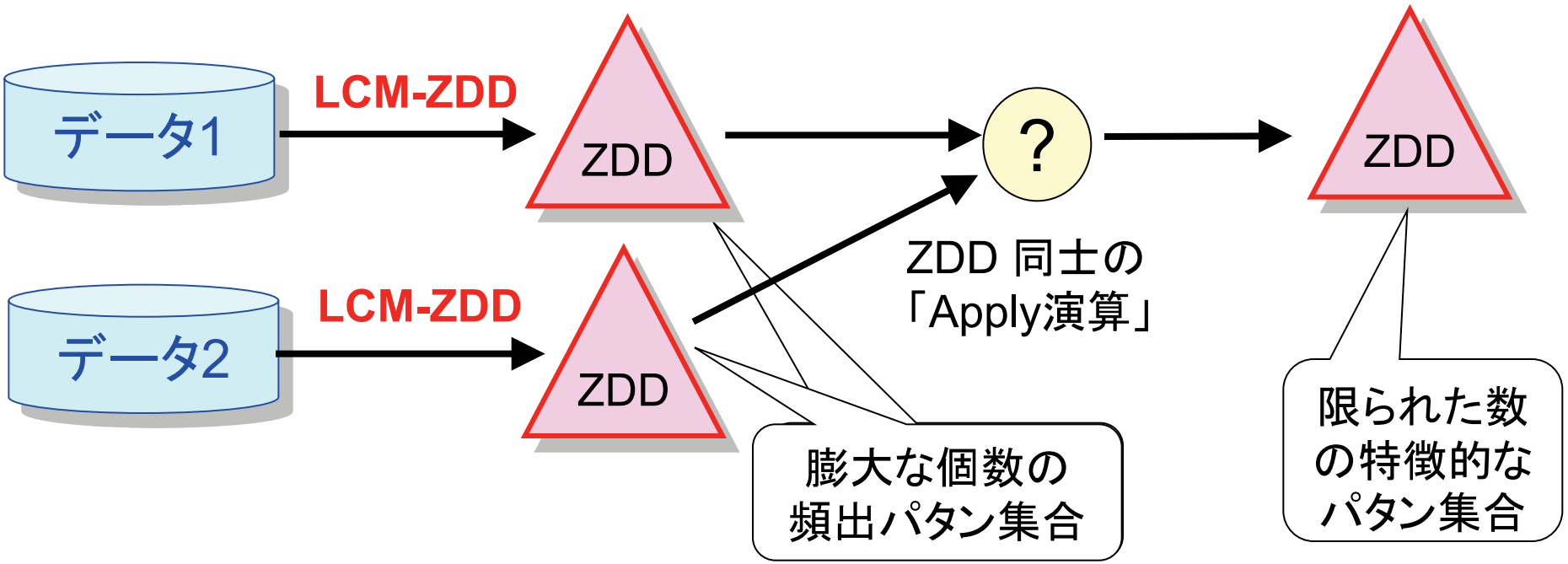
solutions

LCM-ZDD

Original LCM

Dataset name: min. support	# Frequent itemsets	LCMover|ZBDD|	ZBDD Time(s)	LCM-count Time(s)	LCM-dump Time(s)	ZBDD-growth Time(s)
mushroom: 1,000	123,287	760	0.50	0.49	0.64	1.78
500	1,442,504	2,254	1.32	1.30	3.29	3.49
300	5,259,786	4,412	2.25	2.22	9.96	5.11
200	18,094,822	6,383	3.21	3.13	31.63	6.24
100	66,076,586	11,584	5.06	4.87	114.21	6.72
70	153,336,056	14,307	7.16	7.08	277.15	6.97
50	198,169,866	17,830	8.17	7.86	357.27	6.39
T10I4D100K: 100	27,533	8,482	0.85	0.85	0.86	209.82
50	53,386	16,872	0.97	0.92	0.98	242.31
20	129,876	58,413	1.13	1.08	1.20	290.78
10	411,366	173,422	1.55	1.36	1.64	332.22
5	1,923,260	628,491	2.86	2.08	3.54	370.54
3	6,169,854	1,576,184	5.20	3.15	8.14	386.72
2	19,561,715	3,270,977	9.68	5.09	22.66	384.60
BMS-WebView-1: 50	8,192	3,415	0.11	0.11	0.12	29.46
40	48,544	10,755	0.18	0.18	0.22	48.54
36	461,522	28,964	0.49	0.42	0.98	67.16
35	1,177,608	38,164	0.80	0.69	2.24	73.64
34	4,849,466	49,377	1.30	1.07	8.58	83.36
33	69,417,074	59,119	3.53	3.13	144.98	91.62
32	1,531,980,298	71,574	31.90	29.73	3,843.06	92.47
chess: 1,000	29,442,849	53,338	197.58	197.10	248.18	1,500.78
connect: 40,000	23,981,184	3,067	5.42	5.40	49.21	212.84
pumsb: 32,000	7,733,322	5,443	60.65	60.42	75.29	4,189.09
BMS-WebView-2: 5	26,946,004	353,091	4.84	3.62	51.28	118.01

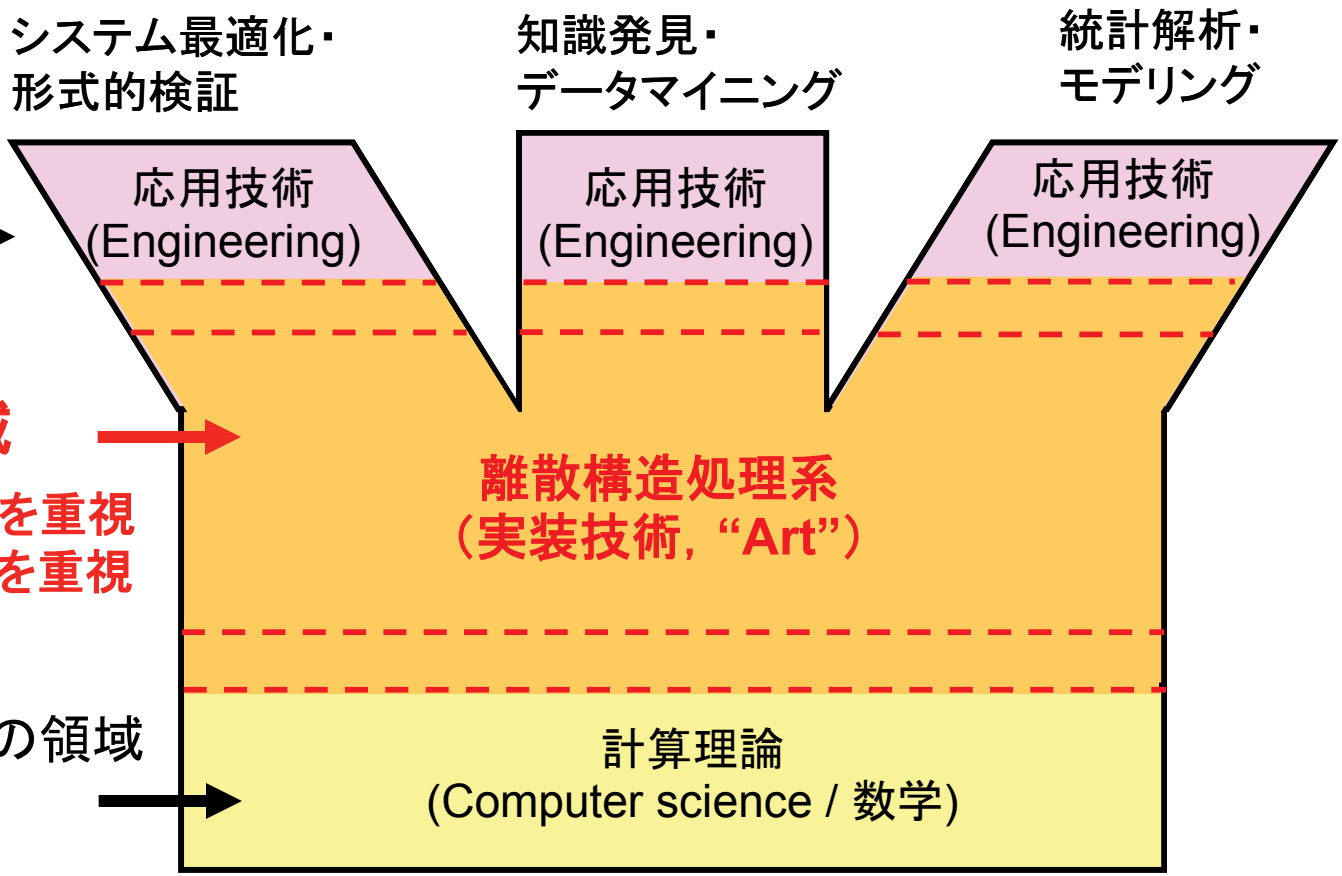
ZDDを用いたパタン集合演算



- 数十億ものパタンを含む集合を圧縮して表現し、ZDDの集合演算を使って効率よく絞込みを行える。
  - 従来の明示的な表現方法では，意味のある解析処理を現実的な時間で行うことは不可能

本研究プロジェクトの対象領域

BDD/ZDD技術の新しい切り口として、様々な離散構造を**統合的に演算処理**する技法を体系化し、**分野横断的かつ大規模な**実問題を高速に処理するための技術基盤を構築する。



個別の工学的応用に特化した技術領域

本研究で扱う技術領域

- **概念・理論だけでなく処理系実装を重視**
- **技術基盤としての簡潔さ・汎用性を重視**

分野横断的な計算理論の領域（概念的・理論的）

本研究プロジェクトの技術面のポイント

現在のZDD

基本的成果

0,1に関して非対称な世界への適用拡大

データマイニング 機械学習

高度な検索　　etc.

ZDDが有効な応用分野が、まだまだ多く残されている。

(組合せ集合モデル)

(より高次のモデル)

- ワード単位データ
- 文字列
- 順列
- 分割
- ツリー
- ネットワーク

より高次の成果

高度なZDD風の離散構造

新しい演算体系の構築

より高次のデータモデルを用いた実用的問題への応用

文字列データ解析

数値データ処理

ツリーや半構造データの解析

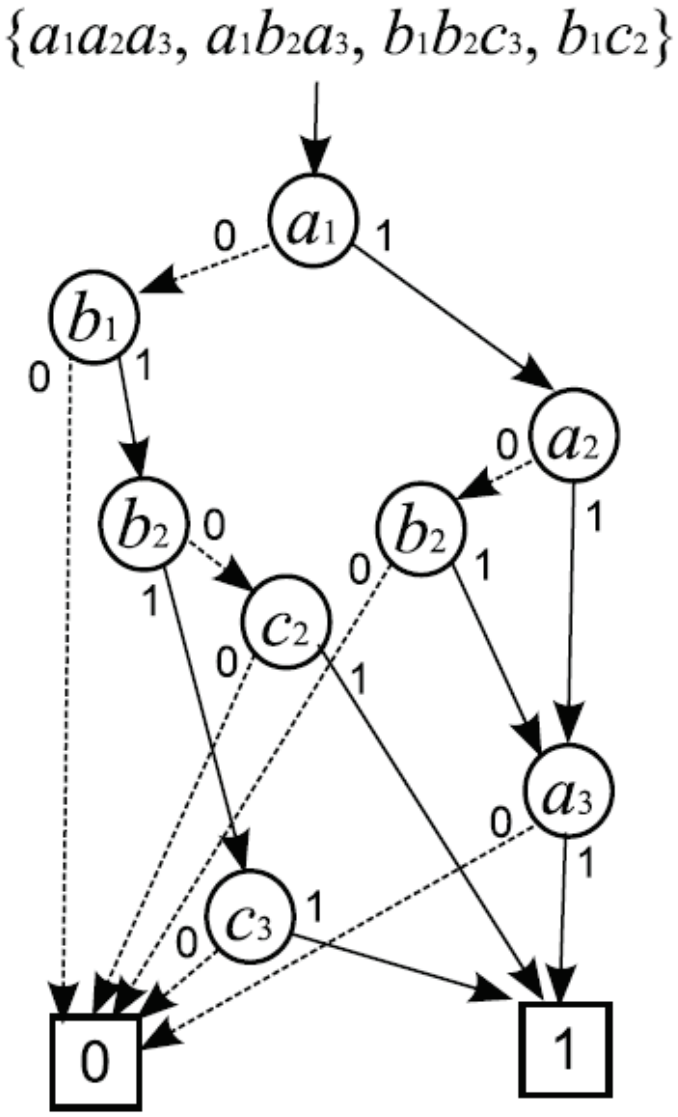
ERATO

Data models and decision diagrams

- **Sets of combinations:** (conventional BDD/ZDD)
  - Don't consider order and duplication of items
  - "abcc" and "bca" are the same.
- **Sets of sequences: (SeqBDD) [Loekito2009]**
  - Distinguishes all finite sequences.
  - φ, {λ}, { ab, aba, bbc }, { a, aa, aaa, aaaa }, etc.
- **Sets of permutations:** (πDD) [Minato2011]
  - Set of orders in a fixed number of items.
  - φ, { abc }, {ab, ba}, { abcdef, acbdfe, bdface }, etc.

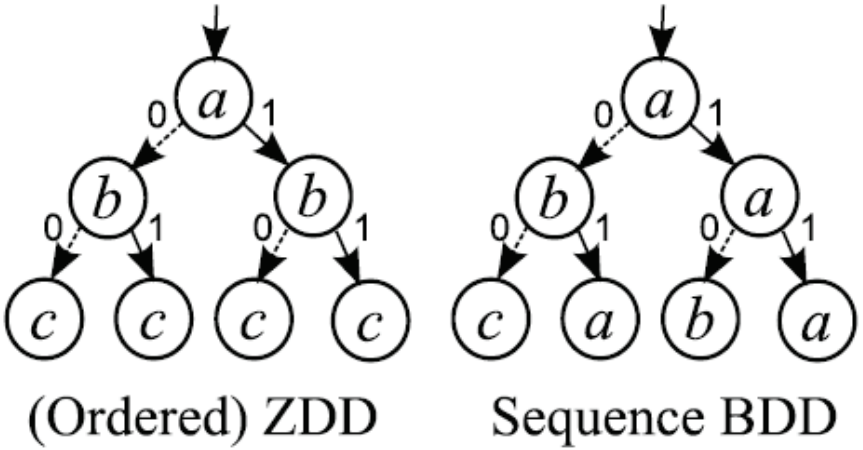
Encoded ZDDs for Sets of sequences

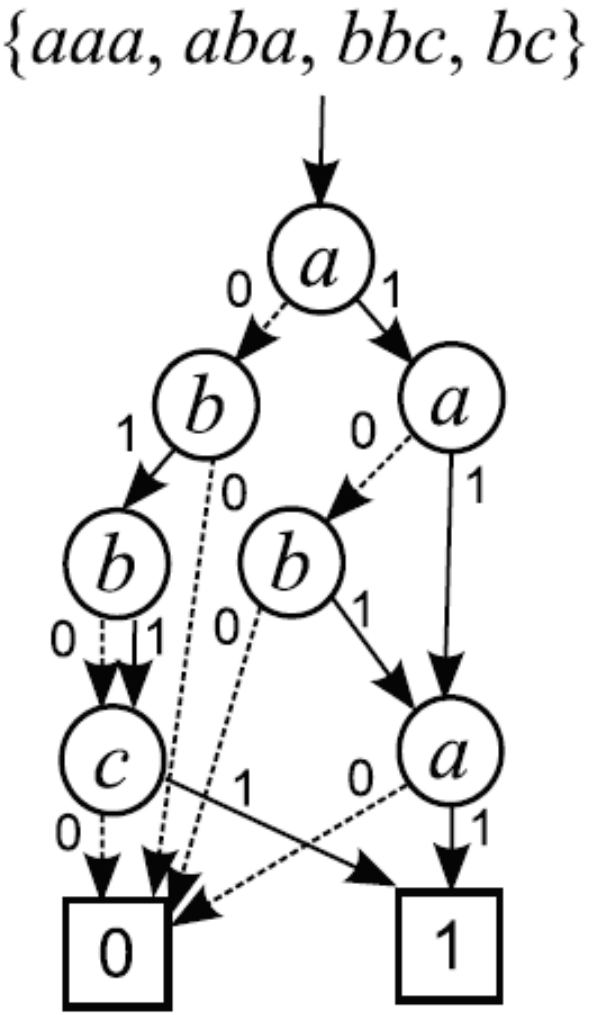
- Pair of (Item - position) is considered different symbol.
  “aaa” → “a1 a2 a3”
  “aba” → “a1 b2 a3”
- Alphabet size: $|\Sigma|$
  Maximum length of sequences: n
  Total encoded symbols: $|\Sigma| \times n$
- Not very efficient.
  - Many symbols needed.
  - We need to put a fixed maximum length of sequences.



Sequence BDD (SeqBDD)

- Loekito, Bailey, and Pei (2009)
  - Same as ZDD reduction rule.
  - **Only 0-edges** keep variable ordering.
  - 1-edges has no restriction.
  - Still unique representation for a given set of sequences.
  - Each path from root to 1-terminal corresponds to a sequence.





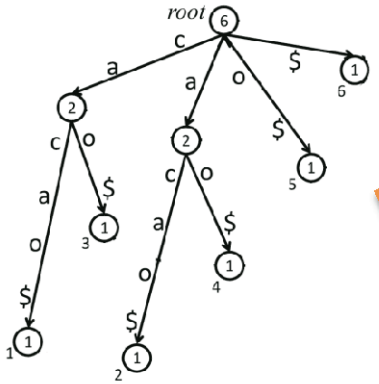
SeqBDDの基本演算 (Sequence Family Algebra)

$\emptyset$	空集合．(0-定数節点を返す.)
$\{\lambda\}$	空列のみからなる集合．(1-定数節点を返す.)
$P.top$	P の最上位の節点の文字 ID を返す.
$P.onset(x)$	文字 x で始まる系列の部分集合を抽出し，各系列の先頭の x を取り除く.
$P.offset(x)$	文字 x 以外で始まる系列の部分集合を抽出する.
$P.push(x)$	P の各系列の先頭に文字 x を付加する.
$P \cup Q$	P と Q の結び (union).
$P \cap Q$	P と Q の交わり (intersection).
$P \setminus Q$	差集合．(P にあって Q にないもの.)
$P.count$	P の要素数を数える.
$P * Q$	P と Q の直積集合 (Cartesian product). P と Q から任意の一つの系列を取り出し連結した系列の集合.

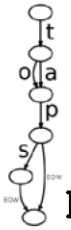
- ZDD風のalgebra
  - *onset*, *offset*, *push* 演算がZDDと異なる。
  - 他の演算はほとんど同じ。

Suffix Tree

[McCreight 1976]



- Substring
- No operations



DAWG

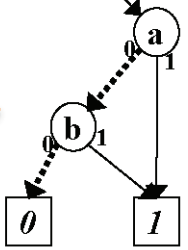
[Blumer 1985]

SeqBDD

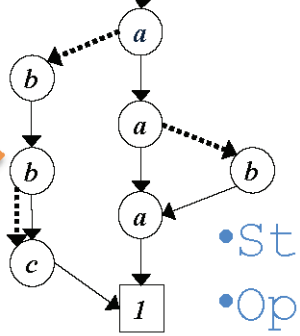
[Loekito 2009]

ZDD

[Minato 1993]

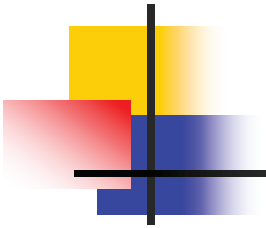


- Combination
- Operations



- String
- Operations

Suffix-DD

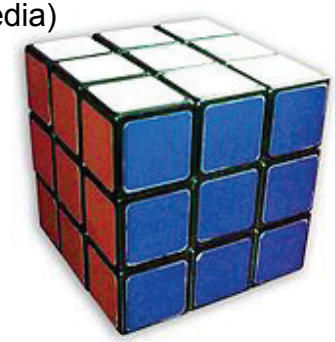


Data models and decision diagrams

- **Sets of combinations:** (conventional BDD/ZDD)
  - Don't consider order and duplication of items
  - "abcc" and "bca" are the same.
- **Sets of sequences:** (SeqBDD) [Loekito2009]
  - Distinguishes all finite sequences.
  - φ, {λ}, { ab, aba, bbc }, { a, aa, aaa, aaaa }, etc.
- **Sets of permutations: (πDD) [Minato2011]**
  - Set of orders in a fixed number of items.
  - φ, { abc }, {ab, ba}, { abcdef, acbdfe, bdface }, etc.

Motivation for developing πDDs

(© Wikipedia)



- Rubik's cube:
  - Let P = { π | any primitive move of cube.}
    → $\boldsymbol{P}$ includes 18 (= 3 ways × 6 faces) permutations.
  - Cartesian product $\boldsymbol{P} \times \boldsymbol{P}$ represents all patterns obtained by twice of primitive moves.
    → $\boldsymbol{P^n}$ will have all patterns by $\boldsymbol{n}$ consecutive moves.
- 15 puzzle / Card games
  - Optimization of packing / arranging strategy
- Primitive sorting networks
  - Classic (and practical) problem considered for long time.
- Design of loss-less codes
  - Any bijective relation corresponds to a permutation.
    → **Useful means to reversible (and quantum) logic design.**

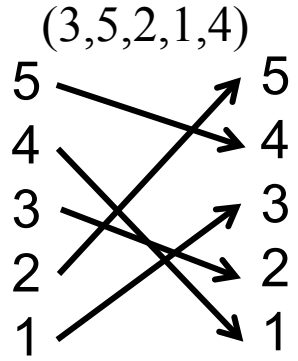
(© Wikipedia)

Decomposition of permutation

- Transpositions $\tau_{(x,y)}$: exchange of two items x and y.
- **Any n-item permutation π can be decomposed by at most $(n-1)$ transpositions.**

Let $x = dim(\pi)$, then $\pi\,\tau_{(x,\,x\pi)}$ must not move x.
Thus, $dim(\pi\,\tau_{(x,\,x\pi)}) \leqq dim(\pi) - 1$

→ Repeating this process provides a decomposed form.



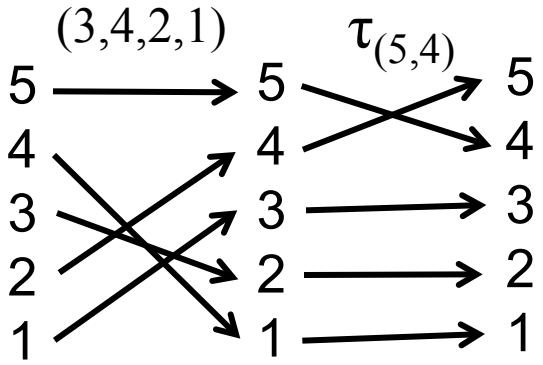
$\pi = (3,5,2,1,4)$

Decomposition of permutation

- Transpositions $\tau_{(x,y)}$: exchange of two items x and y.
- **Any n-item permutation π can be decomposed by at most $(n-1)$ transpositions.**

Let $x = dim(\pi)$, then $\pi\,\tau_{(x,\,x\pi)}$ must not move x.
Thus, $dim(\pi\,\tau_{(x,\,x\pi)}) \leqq dim(\pi) - 1$

→ Repeating this process provides a decomposed form.



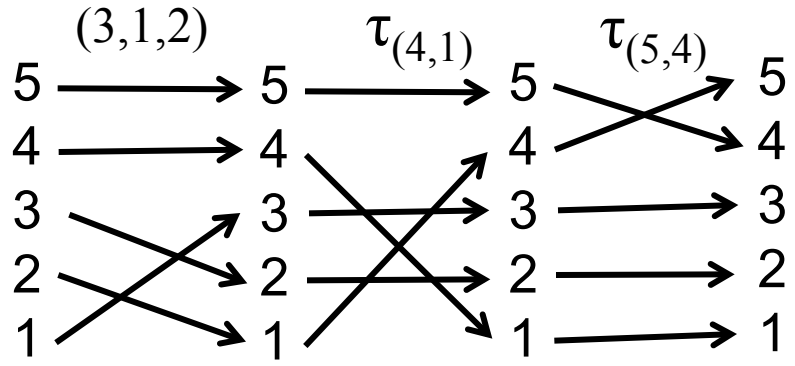
$\pi = (3,5,2,1,4) = (3,4,2,1)\,\tau_{(5,4)}$

Decomposition of permutation

- Transpositions $\tau_{(x,y)}$: exchange of two items x and y.
- **Any n-item permutation π can be decomposed by at most (n-1) transpositions.**

Let $x = dim(\pi)$, then $\pi\,\tau_{(x,\,x\pi)}$ must not move x.
Thus, $dim(\pi\,\tau_{(x,\,x\pi)}) \leqq dim(\pi) - 1$

→ Repeating this process provides a decomposed form.



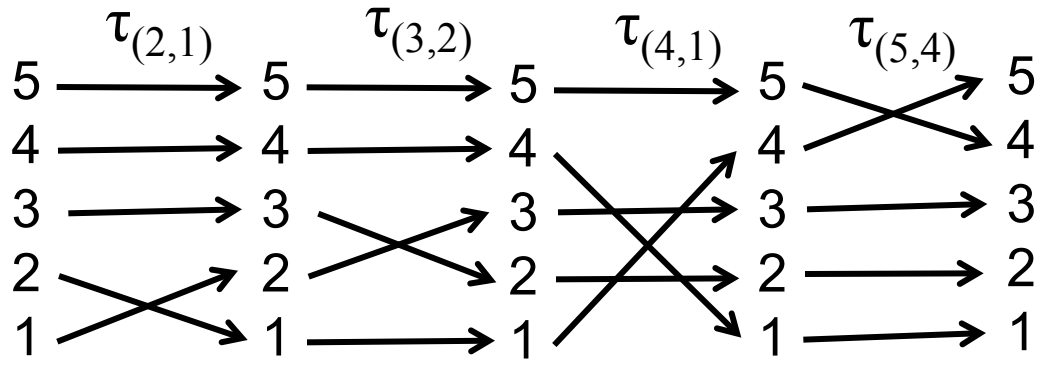
$\pi = (3,5,2,1,4) = (3,1,2)\ \tau_{(4,1)}\ \tau_{(5,4)}$

Decomposition of permutation

- Transpositions $\tau_{(x,y)}$: exchange of two items x and y.
- **Any n-item permutation π can be decomposed by at most $(n-1)$ transpositions.**

Let $x = dim(\pi)$, then $\pi\,\tau_{(x,\,x\pi)}$ must not move x.
Thus, $dim(\pi\,\tau_{(x,\,x\pi)}) \leqq dim(\pi) - 1$

→ Repeating this process provides a decomposed form.

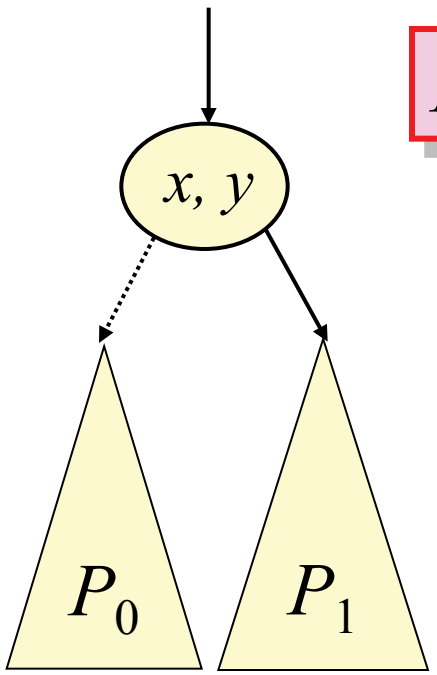


$\pi = (3,5,2,1,4) = \tau_{(2,1)}\,\tau_{(3,2)}\,\tau_{(4,1)}\,\tau_{(5,4)}$

Deterministic process.
→ canonical form for any given π.

Main idea of πDDs

- Using a pair of IDs (x, y) for each decision node.



Let $x = dim(P)$, and $x > y > 0$

$$P = P_0 \cup P_1 \tau_{(x,y)}$$

$$P_0 = \{ \pi \mid \pi \in P, x\pi \neq y \}$$
$$P_1 = \{ \pi\tau_{(x,y)} \mid \pi \in P, x\pi = y \}$$

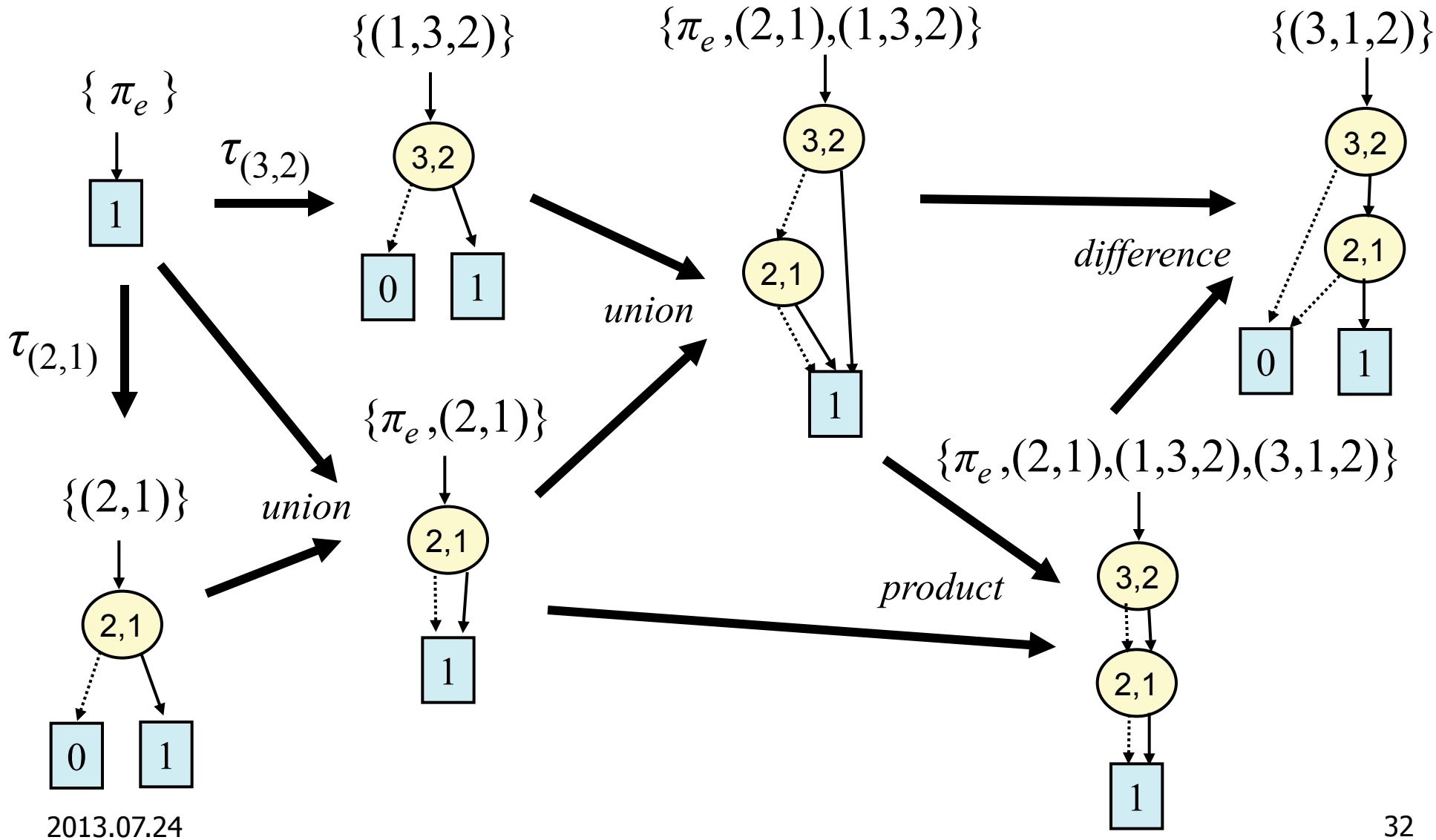
→ $dim(P_0) \leqq dim(P)$
$dim(P_1) < dim(P)$

Algebraic operations for πDDs

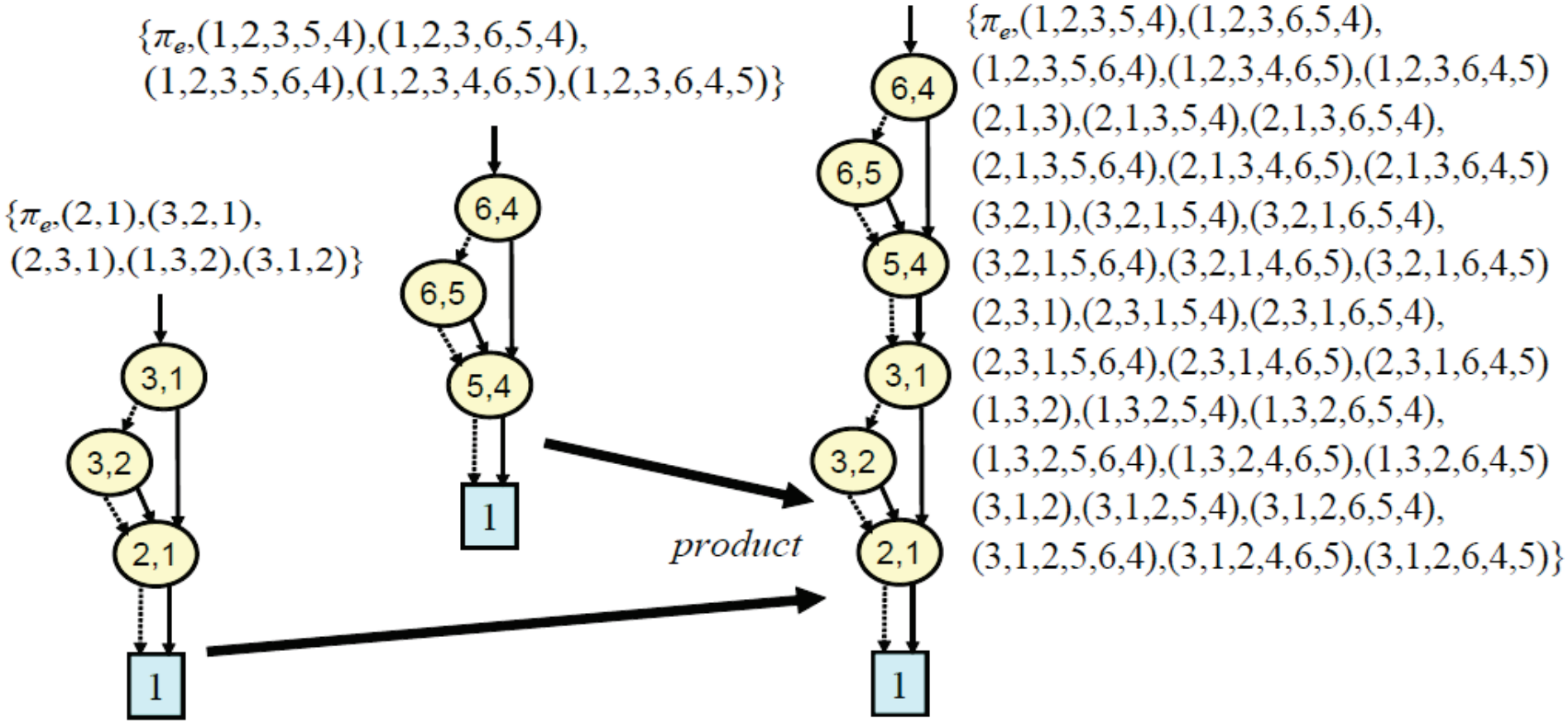
- "Permutation family algebra"

$\emptyset$	Returns the empty set. (0-termial node)
$\{\pi_e\}$	Returns the singleton set. (1-terminal node)
$P.top$	Returns the IDs (x, y) at the root node of P.
$P \cup Q$	Returns $\{\pi \mid \pi \in P \text{ or } \pi \in Q\}$.
$P \cap Q$	Returns $\{\pi \mid \pi \in P,\ \pi \in Q\}$.
$P \setminus Q$	Returns $\{\pi \mid \pi \in P,\ \pi \notin Q\}$.
$P.\tau(x, y)$	Returns $P \cdot \tau_{(x,y)}$.
$P * Q$	Returns $\{\alpha\beta \mid \alpha \in P,\ \beta \in Q\}$.
$P.cofact(x, y)$	Returns $\{\pi\tau_{(x,y)} \mid \pi \in P,\ x\pi = y\}$.
$P.count$	Returns the number of permutations.

Synthesis of πDDs by algebraic operations



Product operation for disjoint permutations



πDD- application for primitive sorting networks

Joint work with Kawahara, Saitoh, and Yoshinaka in ERATO project.

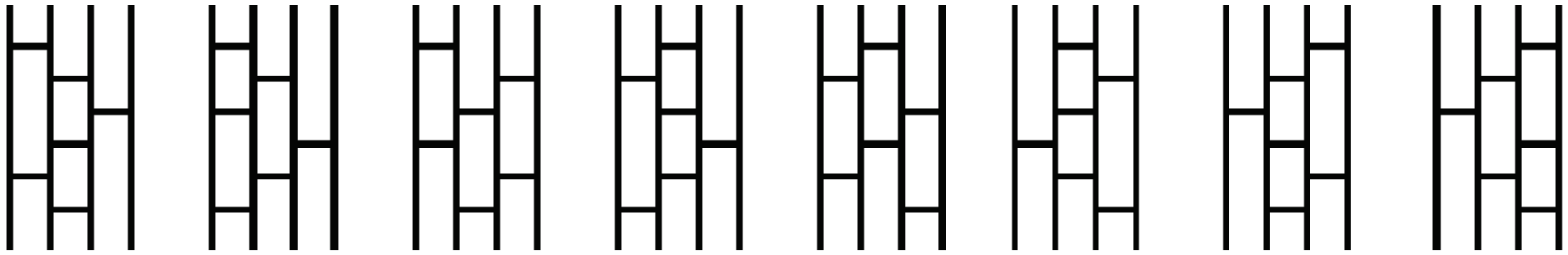


Figure 4: All normal minimum ladders for the reverse permutation when $n = 4$.

- Reverse permutation requires $n(n\text{-}1)/2$ of transpositions.
- **How many irredundant ways to this permutation?**
  - This is a classic problem (solved by Knuth for n=9.)
  - No elegant method. Only enumerative results are known.
  - n=12 was the largest size before our trial.
- **We succeeded in counting the number for n=13.**

日本科学未来館での成果展示

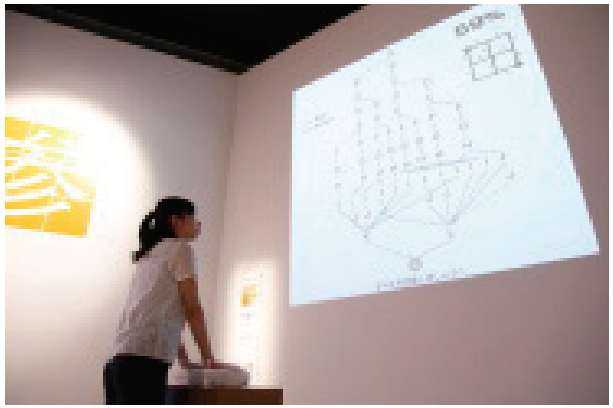
- 日本科学未来館（東京・お台場）で我々の研究プロジェクトの展示「フカシギの数え方」を開催
  - 2012年8月1日～2013年2月25日（期間中、夏休み冬休み1回ずつ）
  - 小中高生・一般市民向けに研究内容をわかりやすく展示
- 好評につき会期延長(~4/25まで)
  - 期間中に23万人が来場
  - ６月以降、北大で再展示

Miraikan

展示の工夫

インタラクティブ展示
（ペンでなぞってみる）

巨大な数表
（視野の広さで実感）

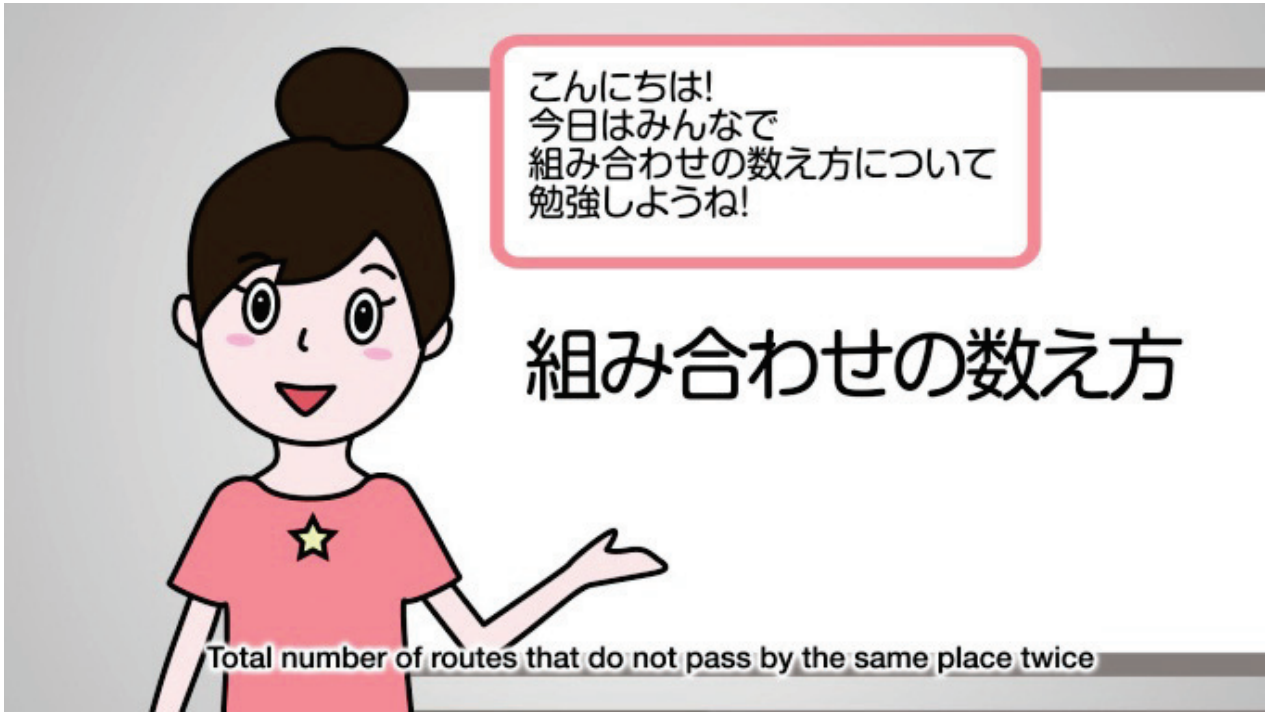


体感展示
（体重をかけて圧縮）

顔写真と説明文
（研究者が語りかける）

アニメーション動画の反響

- 「フカシギの数え方」で、本プロジェクトの企画・監修によるアニメーション動画を製作
- **YouTube再生回数が138万回以上**（ニコ動でも40万回以上）
  - サイエンス系コンテンツでは極めて異例の大ヒットに

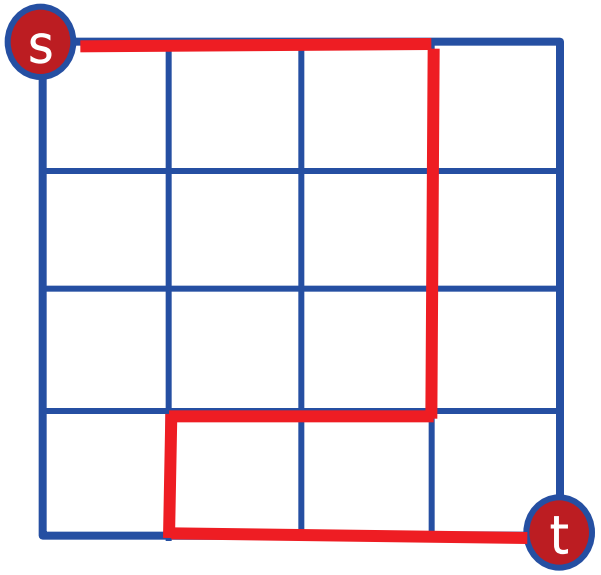


アニメーション動画のねらい

- 組合せ爆発のすごさとアルゴリズム技術の威力を示す
  - 数が大きいだけではなく、計算時間がかかることを実感させる
  - 人間の寿命と対比させる
- 主に小中高・学部生を対象
  - 難しい言葉は使わない
  - SF的なストーリー
- $n \times n$ グリッドグラフを対象とする、
  同じところを2度通らない経路の数え上げ問題
  - 専門家でなくても理解しやすい。（何かに例えなくてもわかる）
  - 一見簡単そうに見える
  - 1，2，3，4，…と淡々と増やしていくと、急激に爆発する

“seif-avoiding walks”の数え上げ

- 最短経路の数え上げは簡単
  (→ $_{2n}C_n$；高校で習う問題)
- 最短でない経路を許すと突然難しくなる。
  （計算式や漸化式は見つかっていない）
  - おねえさんが25万年かかった結果がアニメの中に表示されているため、「計算式教えて」というコメント多数。
  - 残念ながら計算式は知られていない。効率良く数え上げるしかない。



この問題の数え上げ世界記録

- ERATO研究員の岩下氏が**n=21までの数え上げに成功**
  - Knuthのsimpath法を、メモリ効率が良くなるように改善
  - 18×18まではZDDを生成できる
  - 19以降は、ZDDは輪切りにした横一列だけ生成しながら、解の個数だけをカウント。
  - 1節点あたりのビット数も節約して、500GBメモリのマシンで計算に成功（計算時間：約3日）
- Integer Sequences に登録（2012年9月19日）
  - これまではn=19までだった。一気に2段階更新。
  - 登録しようとしたら、9/14の時点でInteger Sequenceに「おねえさん動画」がリンクされていた。（フィンランドの人が提案して認められたらしい）

This site is supported by donations to The OEIS Foundation.

Integer Sequences

Search Hints

(Greetings from The On-Line Encyclopedia of Integer Sequences!)

A007764 Number of nonintersecting (or self-avoiding) rook paths joining opposite corners of an n X n grid. [13]

1, 2, 12, 184, 8512, 1262816, 575780564, 789360053252, 3266598486981642, 41044208702632496804, 156875803046475001321410O, 18241329151424804924147088523 6, 64528039343270018963357185158482118 (list; graph; refs; listen; history; text; internal format)

OFFSET 0,2

COMMENTS
The length of the path varies.
Daniel Forgues, Jan 03 2011 (Start) For n=14, there exists at least one Hamiltonian path from (0,0) to (14,14). For which n do we have at least one Hamiltonian path?
Lattice graphs have their values located at the corners of grid cells. Lattice graph edges join the corners of grid cells.
Grid graphs have their values located at the centers of grid cells. Grid graph edges join the centers of grid cells.
An (m+1) X (n+1) square lattice constitutes the cell corners (coordinates (0,0) to (m,n)) of an m X n square grid.
Number of self-avoiding walks from (0,0) to (n,n), n >= 0, of a (n+1) X (n+1) square lattice.
Since rooks move from centers to centers of adjacent grid cells, should the definition say? "Number of nonintersecting (or self-avoiding) rook paths joining opposite corner cells of an (n+1) X (n+1) grid." (End)

REFERENCES
S. R. Finch, Mathematical Constants, Cambridge, 2003, pp. 331-339.
D. E. Knuth, 'Things A Computer Scientist Rarely Talks About,' CSLI Publications, Stanford, CA, 2001, pages 27-28.
Netnews group rec.puzzles, Frequently Asked Questions (FAQ) file. (Science Section).
D. E. Knuth, The Art of Computer Programming, Section 7.1.4.

LINKS
I. Jensen and H. Iwashita, Table of n, a(n) for n = 0..21 (I. Jensen computed terms 0 to 19)
H. Iwashita, J. Kawahara, and S. Minato, ZDD-Based Computation of the Number of Paths in a Graph
Doi, Maeda, Nagatomo, Niiyama, Sanson, Suzuki, et al, Time with class! Let's count! [Youtube-animation demonstrating this sequence. In Japanese with English translation]
M. Bousquet-Melou, A. J. Guttmann and I. Jensen, Self-avoiding walks crossing a square
S. R. Finch, Self-Avoiding-Walk Connective Constants
I. Jensen, Series Expansions for Self-Avoiding Walks
OEIS Wiki, Self-avoiding walks
Eric Weisstein's World of Mathematics, Self-Avoiding Walk

0	1
1	2
2	12
3	184
4	8512
5	1262816
6	575780564
7	789360053252
8	3266598486981642
9	410442087026324968O4
10	156875803046475001321410O
11	18241329151424804924147088523O
12	64528039343270018963357185158482118
13	69450664761521361664274701548907358996488
14	22744971467681273963182645932798986338761332344O
15	22667455688626727463745673967130989348663248854083190 28
16	68745445609149931587631563132489232824587945968099457285419306
17	63448146112379639713102975407955244004494439868664806936463693878553 36
18	17821128408420651298933849466523252751678380657047676559314524746058266927825 32
19	15233449717048799930807428103192296908994542553232945557760298667373550605928775692558 44
20	39628921998230375602072995171333625021063397057394637715152371133770106823640357067044720649403 98
21	31374751050137102720420538137382214513103312193698723653061351991346433379389385793965576992246021316463868

Previous record:19x19

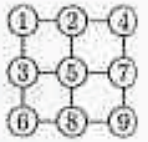
Our new record:21x21

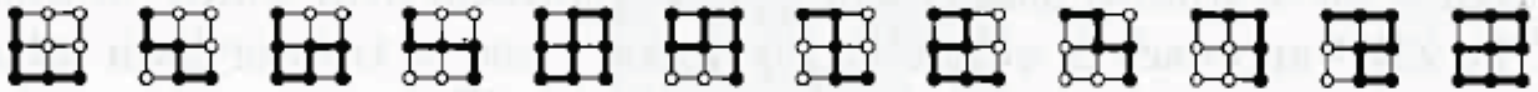
その後の展開

- サ、ツギハ 22×22ネ （by 宇野先生）
  - グラフの形を平面グラフやグリッドグラフに限定すれば、まだ行けそう → 関係者が結集してアルゴリズム開発を続行
- 正月休みに一般市民プログラマからメールが届く
  - 「高速な解き方を思いついたので見てもらえませんか？」
  - 休日に何度か北大に来訪。プロも驚くアイデアを提案
  - 「アマチュアプログラマ」の肩書で共著者に入ってもらう
- 2013年2月：ノルウェーの大学チームが一気にn=24まで記録更新していた。
- 2013年4月：本プロジェクトの総力を結集して**n=25までの計算に成功。再び世界記録を更新。**
  - 主記憶1.5TBの計算機を2～3日占有して計算。
  - 現在のところ、我々が世界記録をキープ。

“simpath” in Knuth-book

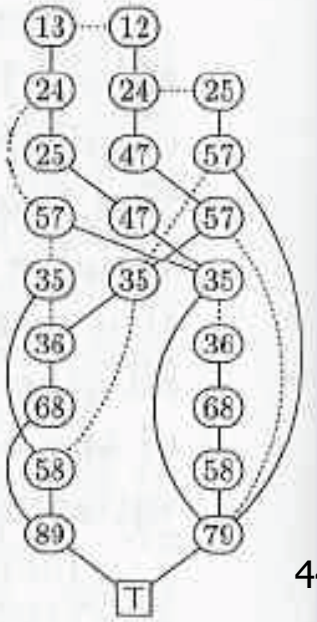
We can also use ZDDs to represent simple paths in an *undirected* graph. For example, there are 12 ways to go from the upper left corner of a 3×3 grid to the lower right corner, without visiting any point twice:





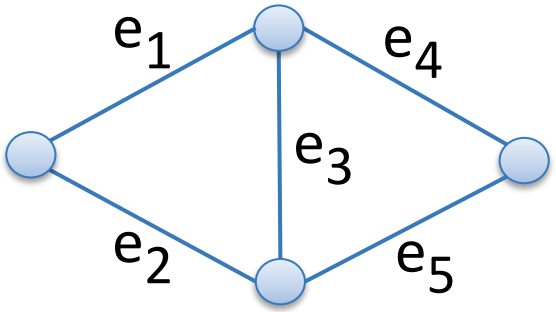
(132)

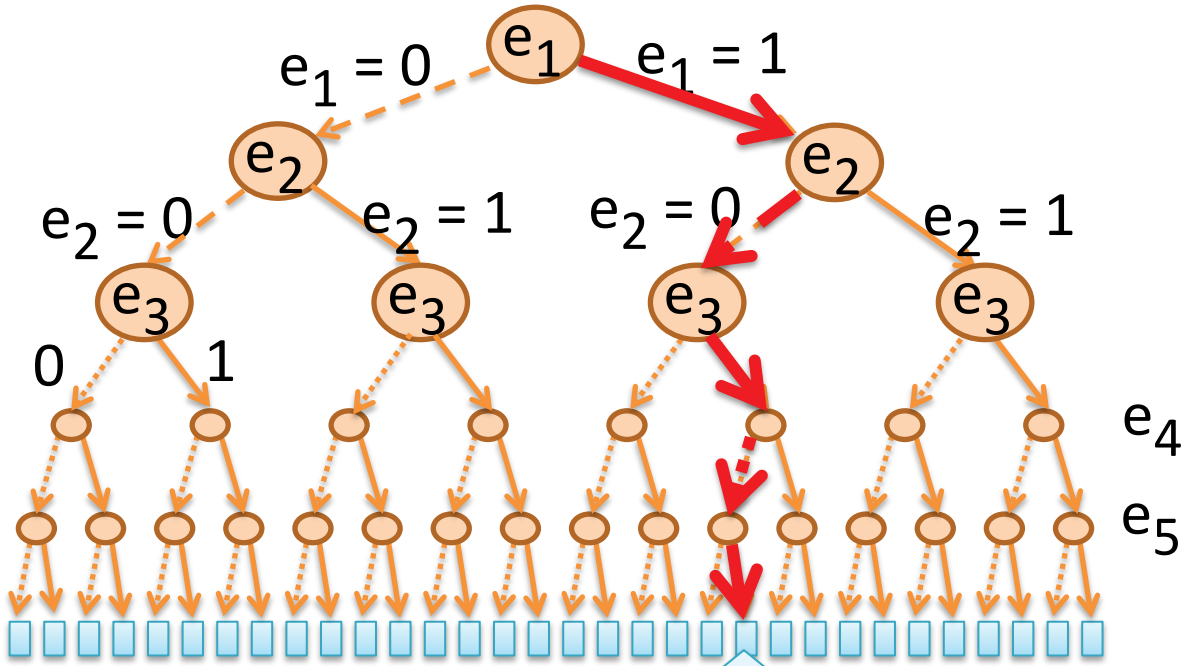
These paths can be represented by the ZDD shown at the right, which characterizes all sets of suitable edges. For example, we get the first path by taking the HI branches at (13), (36), (68), and (89) of the ZDD. (As in Fig. 28, this diagram has been simplified by omitting all of the uninteresting LO branches that merely go to $\boxed{\bot}$.) Of course this ZDD isn't a truly great way to represent (132), because that family of paths has only 12 members. But on the larger grid $P_8 \,\square\, P_8$, the number of simple paths from corner to corner turns out to be 789,360,053,252; and they can all be represented by a ZDD that has at most 33580 nodes. Exercise 225 explains how to construct such a ZDD quickly.



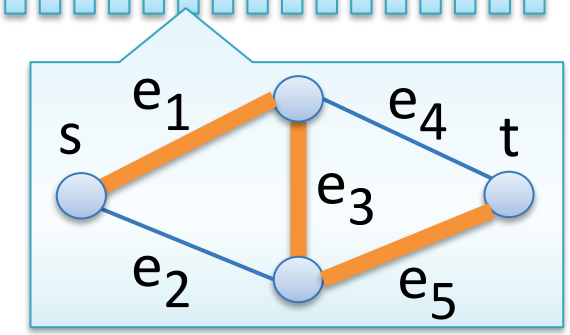
A similar algorithm, discussed in exercise 226, constructs a ZDD that represents all *cycles* of a given graph. With a ZDD of size 22275, we can deduce that $P_8 \,\square\, P_8$ has exactly 603,841,648,931 simple cycles.

BDD/ZDDによるグラフの列挙

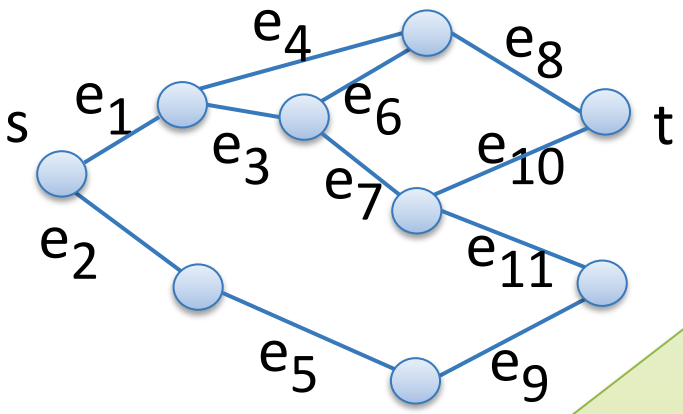


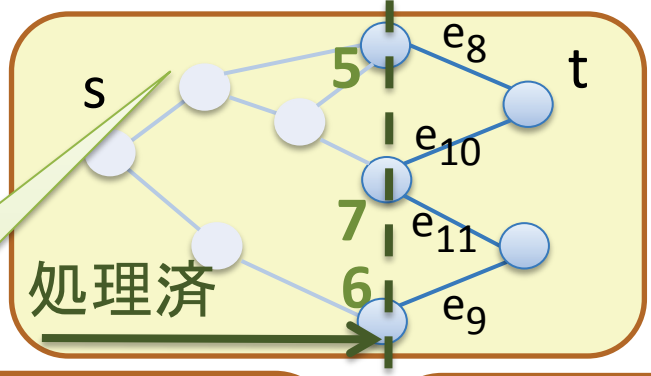


- 与えられた制約を満たすサブグラフ(枝の部分集合)を列挙する問題
  - 各枝の有無が入力変数となる
  - ZDD上での経路がサブグラフに対応
  - 制約を満たす:1-終端、 満たさない:0-終端
- 制約条件を論理式で表せればApply演算でBDD/ZDDを生成できる
  - 部分的に類似する組合せが多数発生する場合、圧縮率が高くなる
  - ある枝を使うと決めたら別の枝が使えなくなることが多い問題ではBDDよりも**ZDDが有利**

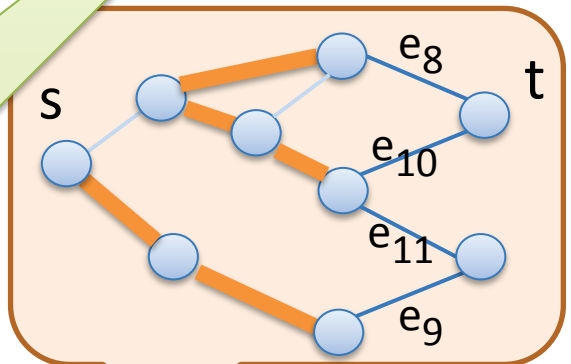


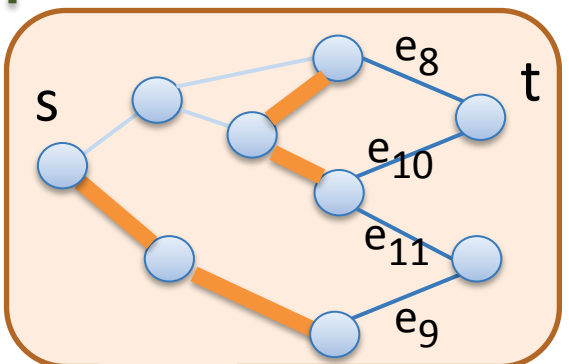
Knuth によるアルゴリズム Simpath

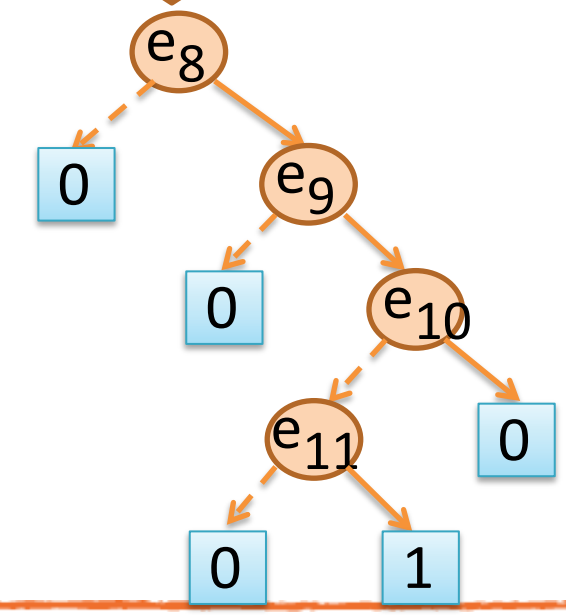


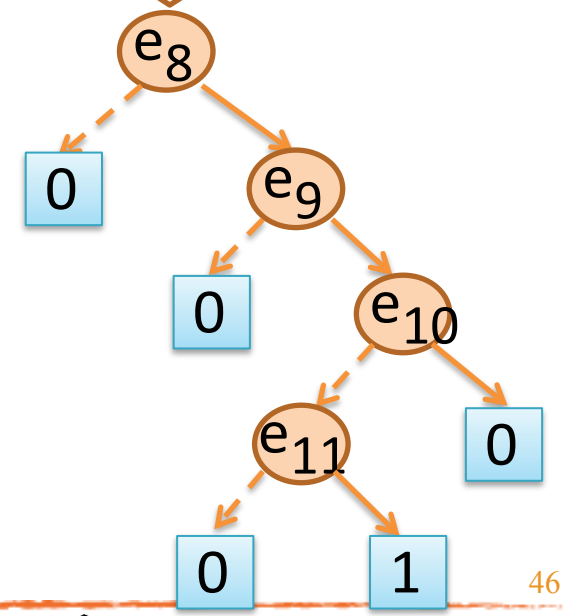


6 は s とつながっている
5 は 7 とつながっている



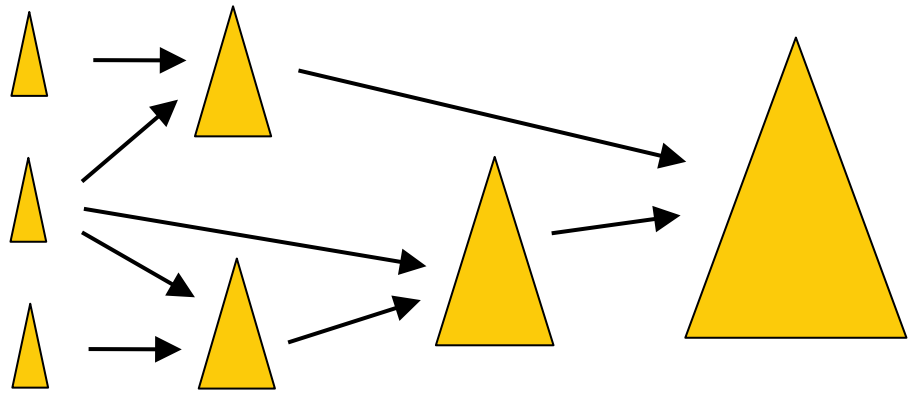




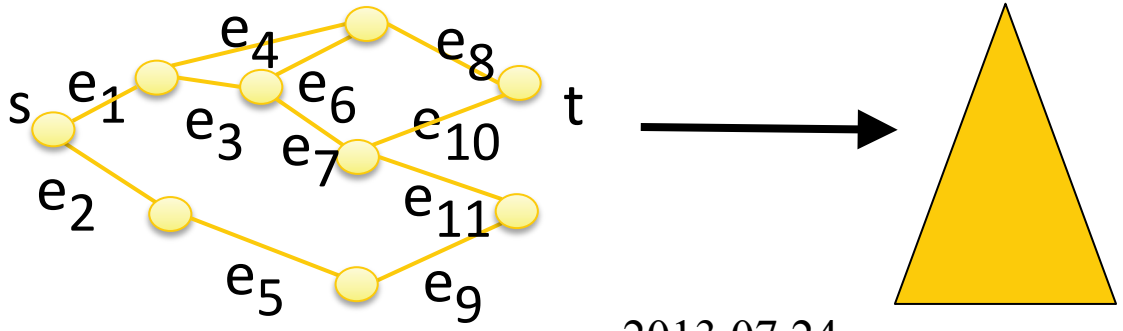


従来のZDD生成とSimpath法の違い

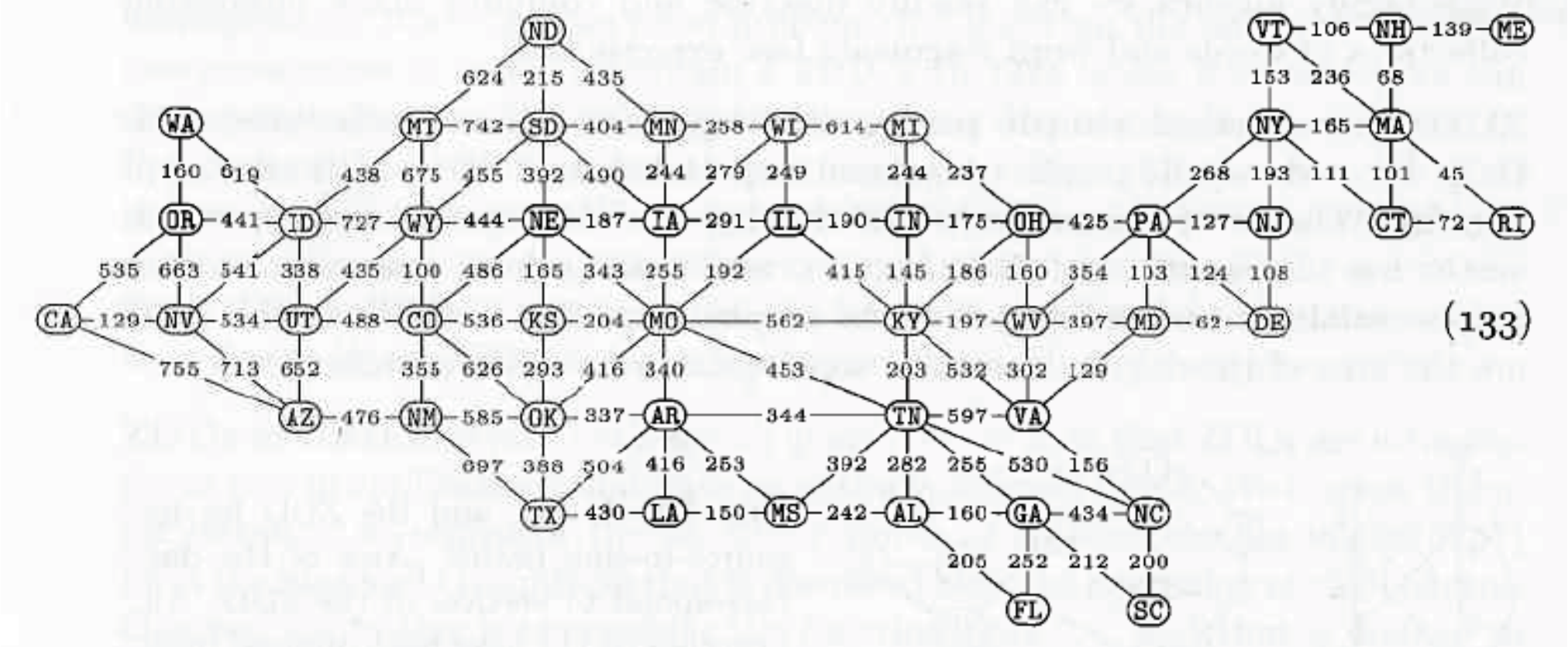
- 従来法：
  ZDD同士の集合演算を何度も適用し、所望のZDDを生成
  - 基本はBryantのApplyアルゴリズム



- Simpath法：
  グラフをたどりながら、最終結果のZDDを一括生成
  - 問題に特有の性質を利用した動的計画法



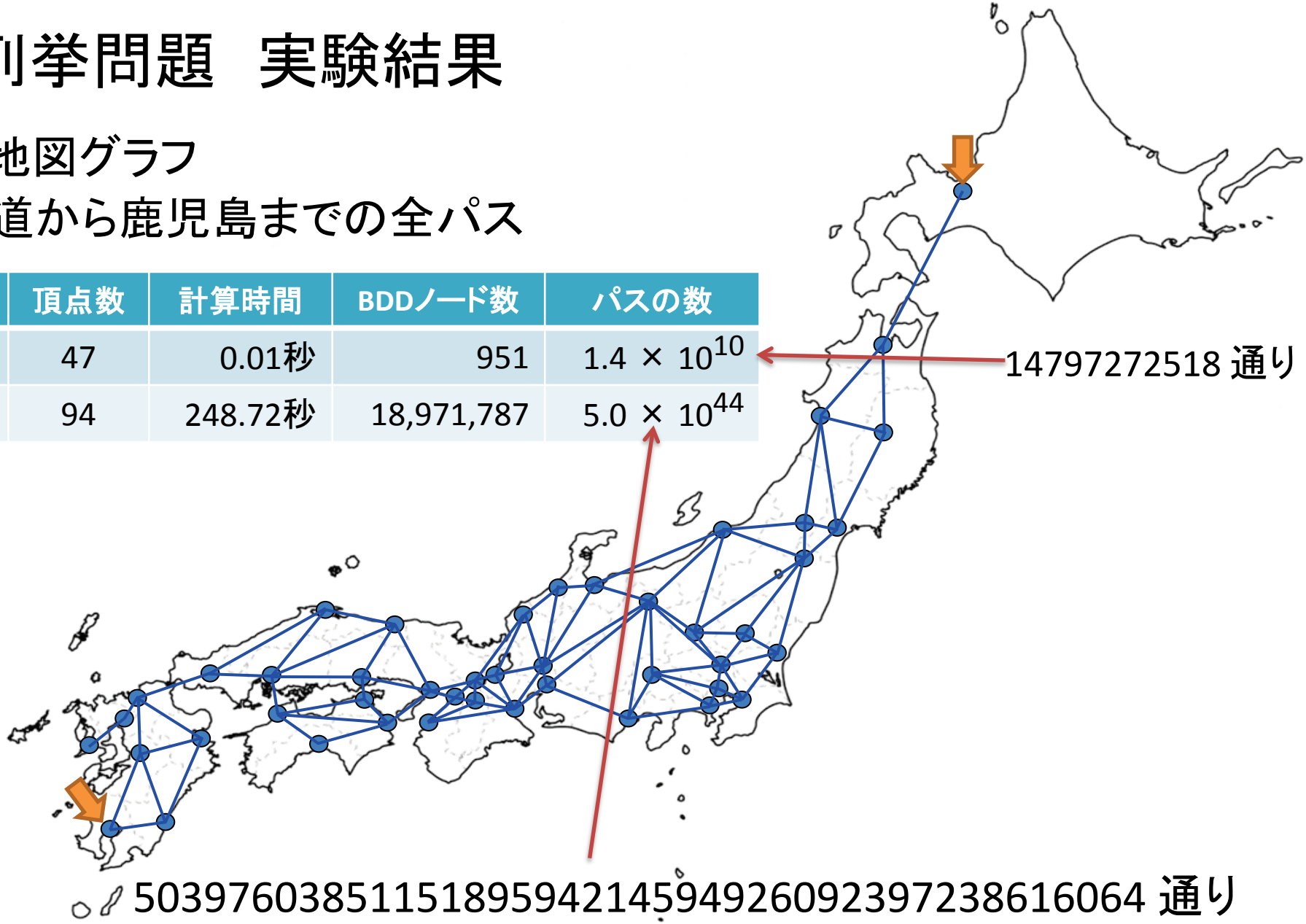
“simpath” for US map in Knuth-book



パス列挙問題　実験結果

日本地図グラフ
北海道から鹿児島までの全パス

	頂点数	計算時間	BDDノード数	パスの数
日本地図	47	0.01秒	951	1.4×10^{10}
2重化	94	248.72秒	18,971,787	5.0×10^{44}



14797272518 通り

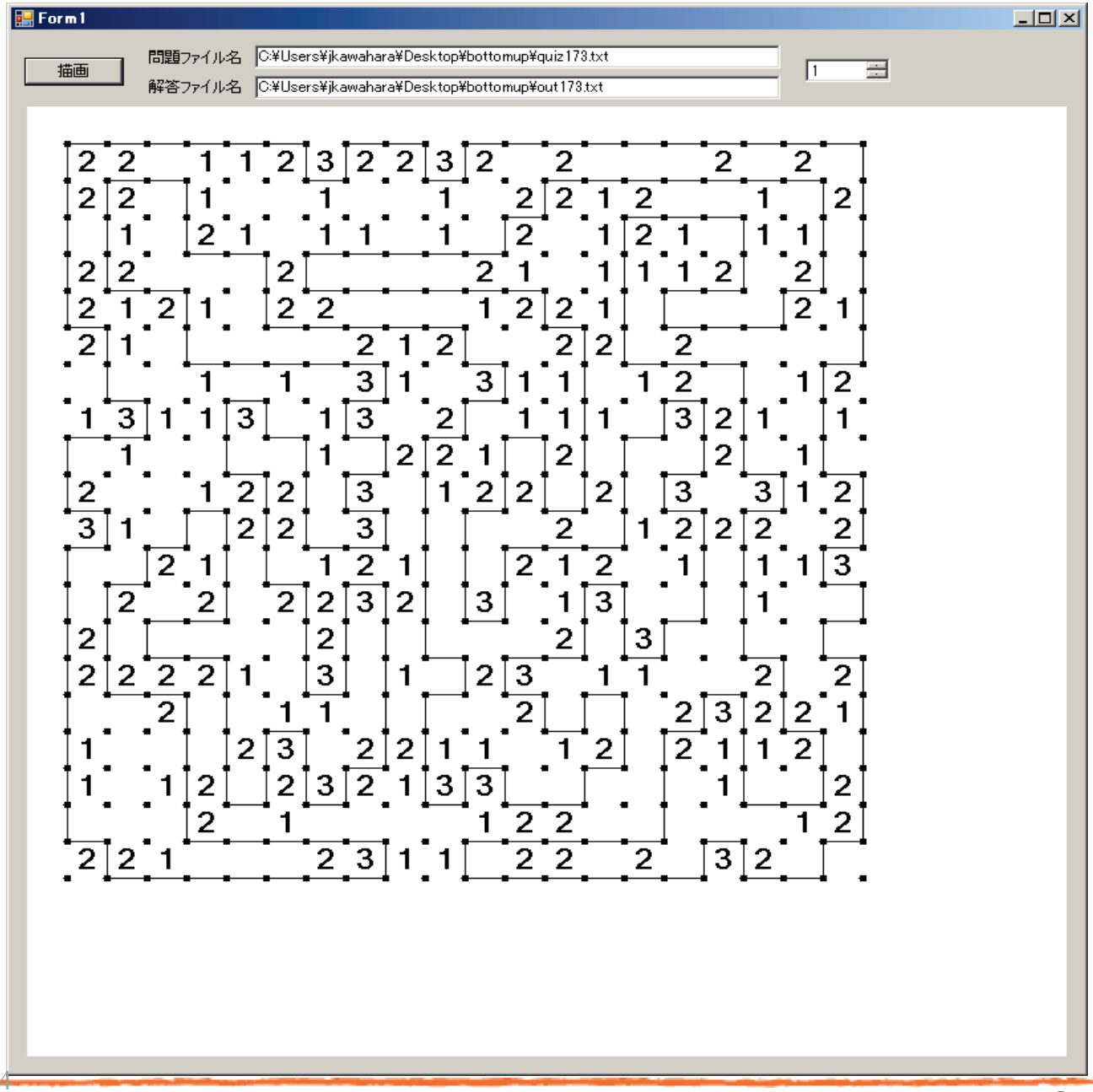
503976038511518959421459492609239723861606 4 通り
（= 503正9760澗3851溝1518穣9594杼2145垓9492京6092兆3972億3861万6064）

フロンティア法の一般化

- パス列挙のバリエーション
  - パス列挙 → サイクル列挙 （Knuth本に演習問題あり）
  - 無向グラフ → 有向グラフでも可能（これも演習問題）
  - 起点終点が複数ペアの場合でも可能（→非交差配線問題）
- ERATOで議論しているうちに、他にも様々なグラフ列挙問題に適用できることが分かってきた
  - 部分木列挙、大域木/林の列挙、カットセット列挙、グラフのk分割問題、連結確率の計算、完全/不完全マッチング列挙、etc.
- Tutte多項式を表すBDD生成法[Sekine-Imai95]との関係
  - さらに調べて行くと、90年代に東大・今井研で、Simpathと極めて類似したアルゴリズムが研究されていたことがわかってきた。
  - ZDDではなくBDDを使用
  - パス列挙ではなく、連結成分の列挙

$$T(x,y) = \sum_{A \subseteq E} (x-1)^{\rho(E)-\rho(A)} (y-1)^{|A|-\rho(A)}$$

条件付きパス(サイクル)の列挙



電力網への応用（昨年度成果プレスリリース）

- 林 泰弘 教授（早稲田大）との共同研究
  - 電力系スマートグリッド業界のリーダ的存在
    （経産省スマートハウス標準化検討会座長、他多数の要職）
  - 電力網最適化の研究で1990年代より湊と協力関係
- 大震災後、より緊急性の高い研究課題に
  - 今後長期的に不足する電力を自然エネルギーで
    補うために必須の電力網解析・制御技術を支援

情報科学の研究者集団として
我が国の苦境を克服するため
できる限り貢献したい。

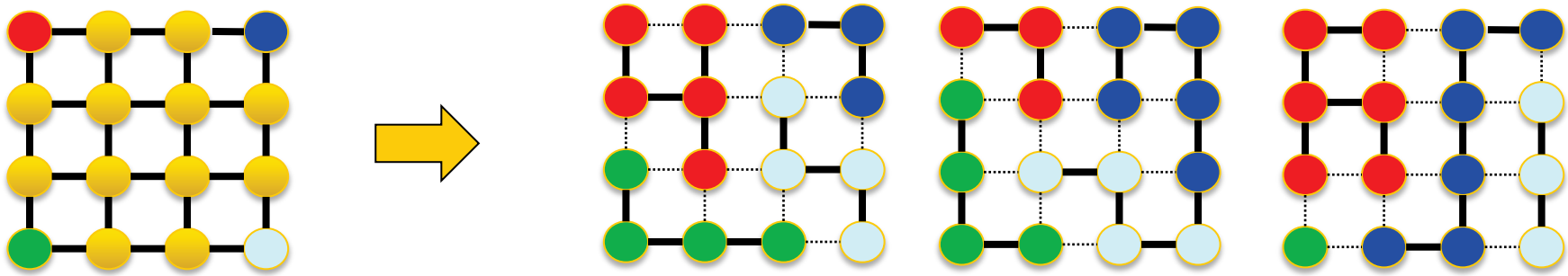
→ ERATOプロジェクト
での取り組みを加速

電力網の問題

- どの変電所からどの領域に給電するか
  - 上手に切り替えれば損失を減らせる
  - 現状では家庭用太陽光発電が普及しても、電力網が不安定になるので、十分に活かせないで捨てている。これをうまく制御して活用できるようにしたい。
  - 災害故障の際にも配電網切り替えの問題が発生する。
- トポロジー制約と電気的制約
  - 開閉器（スイッチ）のON/OFFで制御
  - 各領域はどこか1ヶ所の変電所から給電される。
  - 許容量以上に電流が流れない。
  - 末端電圧が低下しない。
  - できるだけ損失を抑えたい。

グラフk分割問題

- グラフk分割問題（kカットセットの全列挙）
  - 与えられたk頂点を分離するカットセットを全列挙する問題
    →全ての領域がどこか1つの電源から給電される
  - トポロジー制約についてはフロンティア法が使える



- 標準的な電力網モデル（スイッチ468個）で
  トポロジ制約・電気的制約を共に満たすZDDの生成に成功

ZDDノード数:約110万個（779MB）　　実行時間:約1時間15分
解の個数:約10^{63}　（2136那由他8201阿僧祇3834恒河沙8532極9116載8261正2214澗8049溝560穣9817杼8392垓4438京5235兆3981億8952万1540）通り

フロンティア法の工学的応用

社会的に重要な様々な実問題に関係

- 有向/無向グラフのパス列挙：
  - 地理情報システム
  - 大規模システムの依存関係の解析、フローチャートの解析
  - ナンバーリンク、スリザーリンク等のパズル
  - 文字列の連接可能性の列挙
- グラフk分割問題：
  - 電力網の配電区割りの列挙
  - 避難所の配置問題
  - 選挙区割り問題
- 地理情報、電力網、物流網のような社会インフラの構造は，平面グラフや格子グラフに近い形であることが多い。
  → フロンティア法の効果が極めて高くなる傾向がある。

ZDDを生成したら何ができるか

- 解集合を列挙して圧縮して索引化したものがZDD
  - Membership クエリは簡単（上から下にたどるだけ）
  - 解のカウント（ZDDサイズに比例。解の総数よりも圧倒的に高速）
  - あるアイテムを含む（含まない）解集合だけを抽出
  - 抽出した部分集合同士の交わり、結び、差分、
    等価性、包含性を計算
  - 全ての解が、ある条件を満たすかどうかをチェック
  - コスト最小解の探索
  - 上位k個の解の列挙
  - コスト平均値の計算
  - 一様ランダムなサンプリング
- 単なる列挙ではなく索引化している
- 単なる索引ではなくリッチな演算系(algebra)を備えている
- 圧縮が効いている。

最新のプレスリリース(2013.07.23)

ビッグデータから新たな科学的発見をもたらす統計手法を開発

http://www.jst.go.jp/pr/announce/20130723/index.html

- グループリーダの津田さんの成果
  (東工大・瀬々先生らとの共同研究)
- 今週のPNAS(米国科学アカデミー紀要)オンライン速報版に掲載

- 研究成果の概要
  - 従来手法では、特に複合因子の組合せに対して、安全係数のようなものが大き過ぎて悲観的な検定値を出すことが多かった
    (本当は有意かもしれないのに有意でないという結果になっていた)
  - 本手法では、超高速数え上げアルゴリズムを用いることで、格段に精度よく検定できるようになった
  - 今まで論文にならなかった実験結果が論文になる可能性がある
  - 物理学、医学、化学、経済学などあらゆる実験科学に貢献
  - 今後、世界中でものすごく引用される成果になる可能性

まとめ

- 「離散構造処理系」の技術に注目
  - 様々な応用が期待できる基盤技術
- BDD/ZDDの技法に基づく
  - 論理や集合という、もっとも基本的な離散構造モデル
  - さらに高いレベルのデータモデル（系列や順列の集合）に発展
- 最近の成果
  - YouTubeアニメ動画のヒット（のべ再生回数：**138万回**）
  - 「おねえさん問題」として有名に。現在我々が世界記録を保持
  - フロンティア法は様々な実応用を持つ。
    - 電力網, 鉄道, 水道・ガス,
    （地震や津波の）避難所の適正配置, 選挙区の区割り問題
- Graphillionと呼ぶ汎用ソフトウェアツールを開発中
  - 最近公開しました。