ERATO Special Seminar on “Algorithms for Large-Scale Information Processing” (Dec. 2014, Kyoto)
Purpose of this seminar

- Exchange fresh ideas on “Algorithms for Large-Scale Information Processing” by active researchers.
- Inviting two “key researchers” related to this topic.
  - Prof. Adnan Darwiche, UCLA, USA
    - Knowledge compilation.
    - Boolean and probabilistic modeling.
  - Prof. Rajeev Raman, Univ. Leicester, UK
    - Succinct data structures.
    - Bit-string manipulation techniques.
- Show the resent results of “JST ERATO Minato Discrete Structure Manipulation System Project”
  - Aim to international collaboration.
Style of the seminar

- Keynote address by the ERATO leader. (50 min.)
- 2 special invited talks. (60 min. for each)
- 5 selected regular talks (40 min. for each) from the ERATO members and collaborative researchers.
- 9 short position talks (3 min. for each) followed by “poster-session style slide show” and free discussion. (80 min.)
- “Panel-session style discussion” for concluding remarks. (50 min.)
- Excursion to historical and cultural place. (to stimulate exchanging fresh ideas.)
Power of Enumeration — BDD/ZDD-Based Algorithms for Indexing Combinatorial Patterns

Shin-ichi Minato
Hokkaido University, Sapporo, Japan.

ERATO-ALSIP Special Seminar
2014.12.12, Kyoto
Overview of “ERATO” project

- **Top projects** of scientific research in Japan.
  - Executed by JST (Japan Science and Technology Agency).
  - 5 projects / Year are accepted from all scientific subjects. (Computer Science: 0 or 1 project / Year.)
  - 5 year project, total fund: ~10M USD.
  - about 10 PD researchers and 3 admin staffs.

(0th-year) (1st-year) (2nd-year) (3rd-year) (4th-year) (5th-year)

- Acceptance
- Scouting GLs & PDs
- Lab-office construction
- Kick-off Fall WS
- Visitng Knuth’s home
- AMBN 2010
- Visiting CMU
- Negotiation to collab with company
- Spring WS
- FIT2010 Special sympo.
- Special sympo.
- Visiting CMU
- ALSIP 2011
- Power NW Press rel.
- Shin-ichi Minato
- Spring WS
- Fall WS
- Spring WS
- Special issue
- JSAI Journal
- Miraikan exhibition
- ALSIP 2012
- Spring WS
- Fall WS
- Spring WS
- FIT2013 sympo.
- HU-museum exhibition
- RM 2013
- “LAMP” Press rel.
- Project report
- RC 2013
- ALSIP 2014
Our main subject

Discrete structures and applications

Many problems solved by computers can be decomposed as a type of **discrete structures** using simple primitive operations. → Often needs **a huge amount of enumerative operations**.

<table>
<thead>
<tr>
<th>design automation</th>
<th>data mining / knowledge discovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>fault analysis</td>
<td>bio informatics</td>
</tr>
<tr>
<td>constraint satisfaction problem</td>
<td>machine learning / classification</td>
</tr>
<tr>
<td>web data analysis</td>
<td></td>
</tr>
</tbody>
</table>

**Discrete structure manipulation system**

- set theory
- symbolic logic
- inductive proof
- combinatorics
- graph theory
- probability theory

So many applications → Important for the society.

Performance Improvement (10–100x)

Foundational materials for C.S. and math.
BDD (Binary Decision Diagram)

- Developed in VLSI CAD area mainly in 1990’s.

Canonical form for given Boolean functions under a fixed variable ordering.

Node elimination rule

Node sharing rule

2014.12.12 Shin-ichi Minato
Effect of BDD reduction rules

- Exponential advantage can be seen in extreme cases.
  - Depends on instances, but effective for many practical ones.
BDD-based logic operation algorithm

- If the BDD starting from the binary tree: always requires exponential time & space.
- Innovative BDD synthesis algorithm
  - Proposed by R. Bryant in 1986.
  - Best cited paper for many years in all EE&CS areas.

A BDD can be constructed from the two operands of BDDs. (Computation time is almost linear for BDD size.)
Boolean functions and sets of combinations

Boolean function:
\[ F = (a \land b \lor \neg c) \lor (\neg b \land c) \]

Set of combinations:
\[ F = \{ab, ac, c\} \]

Operations of combinatorial itemsets can be done by BDD-based logic operations.
- Union of sets \( \rightarrow \) logical OR
- Intersection of sets \( \rightarrow \) logical AND
- Complement set \( \rightarrow \) logical NOT

(customer’s choice)
Zero-suppressed BDD (ZDD) [Minato93]

- A variant of BDDs for sets of combinations.
- Uses a new reduction rule different from ordinary BDDs.
  - Eliminate all nodes whose “1-edge” directly points to 0-terminal.
  - Share equivalent nodes as well as ordinary BDDs.
- If an item $x$ does not appear in any itemset, the ZDD node of $x$ is automatically eliminated.
  - When average occurrence ratio of each item is 1%, ZDDs are more compact than ordinary BDDs, up to 100 times.
The latest Knuth’s book fascicle (Vol. 4-1) includes a BDD section with 140 pages and 236 exercises.

In this section, Knuth used 30 pages for ZDDs, including more than 70 exercises.

- I honored to serve proofreading of the draft version of his article.
- Knuth recommended to use “ZDD” instead of “ZBDD.”
- He reorganized ZDD operations and named “Family Algebra.”
- 2010/05, I visited Knuth’s home and discussed the direction of future work.
Algebraic operations for ZDDs

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

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

Formerly I called this “unate cube set algebra,” but Knuth reorganized as “Family algebra.”

Useful for many practical applications.

2014.12.12 Shin-ichi Minato
 Applications of BDDs/ZDDs

- BDD-based algorithms have been developed mainly in VLSI logic design area. (since early 1990’s.)
  - Equivalence checking for combinational circuits.
  - Symbolic model checking for logic / behavioral designs.
  - Logic synthesis / optimization.
  - Test pattern generation.

- Recently, BDDs/ZDDs are applied for not only VLSI design but also for more general purposes.
  - Data mining (Fast frequent itemset mining) [Minato2008]
  - Text analysis (manipulating “sets of sequences”) using Sequence BDDs [Loekito2009]
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.
    (considers positions and duplications)
  - $\phi$, \{\lambda\}, \{ab, aba, bbc\}, \{a, aa, aaa, aaaa\}, etc.

- **Sets of permutations:** ($\pi$DD) [Minato2011]
  - Set of orders in a fixed number of items.
  - $\phi$, \{123\}, \{12, 21\}, \{123456, 132456, 246135\}
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. (considers positions and duplications)
  - \(\emptyset, \{\lambda\}, \{ab, aba, bbc\}, \{a, aa, aaa, aaaa\}\), etc.

- **Sets of permutations:** (πDD) [Minato2011]
  - Set of orders in a fixed number of items.
  - \(\emptyset, \{123\}, \{12, 21\}, \{123456, 132456, 246135\}\)
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.

(ordered) ZDD

Sequence BDD
Basic operations of sequence family algebra

- ZDD-like algebraic operations.
  - \textit{onset}, \textit{offset}, and \textit{push} operations are different.
  - Other operations are almost same.

| \(\emptyset\)  | Returns empty set. (0-terminal node) |
| \(\{\lambda\}\) | Returns the set of only null-combination. (1-terminal node) |
| \(P.\text{top}\) | Returns item-ID at root node of \(P\). |
| \(P.\text{onset}(x)\) | Selects the subset of sequences that begin with letter \(x\), and then removes \(x\) from the head of each sequence. |
| \(P.\text{offset}(x)\) | Selects the subset of sequences that do not begin with letter \(x\). |
| \(P.\text{push}(x)\) | Appends \(x\) to the head of every sequence in \(P\). |
| \(P \cup Q\) | Returns union set. |
| \(P \cap Q\) | Returns intersection set. |
| \(P \setminus Q\) | Returns difference set. (in \(P\) but not in \(Q\).) |
| \(P.\text{count}\) | Counts number of combinations. |
| \(P \times Q\) | Cartesian product of \(P\) and \(Q\). (Concatenations of all pairs in \(P\) and in \(Q\) ) |
Suffix Tree
[McCreight 1976]

DAWG
[Blumer 1985]

• Substring
• No operations

SeqBDD
[Loekito 2009]

ZDD
[Minato 1993]

• Combination
• Operations

Suffix-DD

• String
• Operations

String
Sets
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.
    (considers positions and duplications)
  - $\emptyset$, $\{\lambda\}$, $\{ab, aba, bbc\}$, $\{a, aa, aaa, aaaa\}$, etc.

- **Sets of permutations:** ($\pi$DD) [Minato2011]
  - Set of orders in a fixed number of items.
  - $\emptyset$, $\{123\}$, $\{12, 21\}$, $\{123456, 132456, 246135\}$
Motivation for considering πDDs

- Rubik’s cube: Let \( P = \{ \pi | \text{any primitive move of cube.} \} \)
  - \( P \) includes 18 (= 3 ways \( \times \) 6 faces) permutations.
  - Cartesian product \( P \times P \) represents all possible patterns obtained by twice of primitive moves.
  - \( P^{20} \) will have all possible patterns. (but maybe too large.)

- 15 puzzle / Card games
  - Optimization of packing / arranging strategy

- “Amida”-drawing (primitive sorting networks)
  - One-to-one matching problems between two parties.
  - Any bijective relation corresponds to a permutation.

- Design of loss-less codes.
  - Analysis of reversible logic. (related to quantum logic circuit.)
Decomposition of permutation

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

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

→ 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 $\pi$. 

2014.12.12
Shin-ichi Minato 24
Main idea of $\pi$DDs

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

Let $x = \text{dim}(P)$, and $x > y > 0$

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

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

$\Rightarrow$

$\text{dim}(P_0) \leq \text{dim}(P)$
$\text{dim}(P_1) < \text{dim}(P)$
\(\pi\)DDs for sets of permutations

\[
\{(2,1)\}
\]

\[
\{\pi_e, (2,1)\}
\]

\[
\{(2,1),(3,1,2),(3,2,1),(2,3,1)\}
\]

\[
\{\pi_e, (2,1,),(1,3,2),(3,1,2)\}
\]

\[
\{(2,1),(3,1,2),(3,2,1),(2,3,1)\}
\]

\[
\{\pi_e, (2,1), (1,3,2),(3,1,2)\}
\]

\[
\{(2,1),(3,1,2),(3,2,1),(2,3,1)\}
\]

\[
\{\pi_e, (2,1)\}
\]

\[
\{\pi_e, (2,1)\}
\]
Product operation for disjoint permutations

\{e, 123546, 123654, 123564, 123465, 123645\}

\{e, 213, 321, 231, 132, 312\}
Algebraic operations for $\pi$DDs

- “Permutation family algebra”

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>Returns the empty set. (0-terminal node)</td>
</tr>
<tr>
<td>${\pi_e}$</td>
<td>Returns the singleton set. (1-terminal node)</td>
</tr>
<tr>
<td>$P.\text{top}$</td>
<td>Returns the IDs $(x, y)$ at the root node of $P$.</td>
</tr>
<tr>
<td>$P \cup Q$</td>
<td>Returns ${\pi \mid \pi \in P \text{ or } \pi \in Q}$.</td>
</tr>
<tr>
<td>$P \cap Q$</td>
<td>Returns ${\pi \mid \pi \in P, \pi \in Q}$.</td>
</tr>
<tr>
<td>$P \setminus Q$</td>
<td>Returns ${\pi \mid \pi \in P, \pi \notin Q}$.</td>
</tr>
<tr>
<td>$P.\tau(x, y)$</td>
<td>Returns $P \cdot \tau(x, y)$.</td>
</tr>
<tr>
<td>$P \ast Q$</td>
<td>Returns ${\alpha \beta \mid \alpha \in P, \beta \in Q}$.</td>
</tr>
<tr>
<td>$P.\text{cofact}(x, y)$</td>
<td>Returns ${\pi \tau(x, y) \mid \pi \in P, x\pi = y}$.</td>
</tr>
<tr>
<td>$P.\text{count}$</td>
<td>Returns the number of permutations.</td>
</tr>
</tbody>
</table>
Technical direction of our project

Current ZDDs

Applications in asymmetric world
- Data mining, Machine learning
- Advanced searching etc.

(Combinatorial)

Applications with higher data model
- Sequence data analysis
- Numerical data processing
- Processing of trees or semi-structured data

(Higher model)

Further outputs

Advanced ZDD-like structure

Develop special new algebraic operations.

- Multisets
- Sequences
- Permutations
- Partitions
- Trees, DAGs
- Networks
- etc.

Still many applications remain where ZDDs would be effective.
Exhibition at National Science Museum

"Billion years reduced into a few seconds" by state-of-the-art ALG.

1,594,909 views on YouTube!

Total number of routes that do not pass by the same place twice
Purpose of the movie

- Shows strong power of combinatorial explosion, and importance of algorithmic techniques.
- Mainly for junior high school to college students
  - Not using any difficult technical terms.
  - Something like a funny science fiction story.
- We used the enumerating problem of “self-avoiding walk” on $n \times n$ grid graphs
  - This problem is discussed in the ZDD-section of the Knuth-book, section 7.1.4.
  - We received a letter from Knuth:
    “I enjoyed the You-Tube video about big numbers, and shared it to several friends.”
Enumerating “self-avoiding walks”

- Counting shortest s-t paths is quite easy. ($\rightarrow 2^n C_n$; educated in high school.)
- If allowing non-shortest paths, suddenly difficult. No simple calculation formula has been found.
  - Many people requested the formula because the movie shows a super-big number, which the teacher spent 250,000 years to count.
  - However, no formula exists. Only efficient algorithm!
Integer Sequences: A007764

I. Jensen, H. Iwashita, R. Spaans, Table of n, a(n) for n = 0..25 (I. Jensen computed terms 0 to 19, H. Iwashita computed 20 and 21, R. Spaans computed 22 to 24, and H. Iwashita computed 25)
M. Bousquet-Melou, A. J. Guttmann and I. Jensen, Self-avoiding walks crossing a square
Doi, Maeda, Nagatomo, Niiyama, Sanson, Suzuki, et al., Time with class! Let's count! [Youtube-animation demonstrating this sequence. In Japanese with English translation]
S. R. Finch, Self-Avoiding-Walk Connective Constants
H. Iwashita, J. Kawahara, and S. Minato, ZDD-Based Computation of the Number of Paths in a Graph
H. Iwashita, Y. Nakazawa, J. Kawahara, T. Uno, and S. Minato, Efficient Computation of the Number of Paths in a Grid Graph with Minimal Perfect Hash Functions
I. Jensen, Series Expansions for Self-Avoiding Walks
OEIS Wiki, Self-avoiding walks
Ruben Grönning Spaans, C program
Eric Weisstein's World of Mathematics, Self-Avoiding Walk

EXAMPLE
Suppose we start at (0,0) and end at (n-1,n-1). Let U, D, L, R denote steps that are up, down, left, right.

a(2) = 2: UR or RU.
a(3) = 12: UURR, UURDRU, UURDDU, URUR, URRU, URDRUU and their reflections in the x=y line.

CROSSES
Sequence in context: A134716 A006023 A039748 * A015195 A051421 A182162
Adjacent sequences: A007761 A007762 A007763 * A007785 A007766 A007767

KEYWORD
nonn,walk,hard,more,nice

AUTHOR
David Radcliffe and D. E. Knuth.

EXTENSIONS
Extended to n=19 by M. Bousquet-Melou, A. J. Guttmann and I. Jensen
Extended to n=21 using ZDD technique based on Knuth's The Art of Computer Programming (exercise 225 in 7.1.4) by H. Iwashita, J. Kawahara, and S. Minato, Sep 18 2012
Extended to n=24 using state space compression (with rank/unrank) and dynamic programming (based in I. Jensen) by Ruben Grönning Spaans, Feb 22 2013
Extended to n=25 by Hiroaki Iwashita, Apr 11 2013
26 x 26: Our record in Dec 2013
(1404 edges included in the graph.)
(⇒ 2^{1404} combinations of edges.)

The last record was 24 x 24 by a group in Norway.
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 $3 \times 3$ grid to the lower right corner, without visiting any point twice:

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 $\bot$.) 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.
Knuth’s algorithm “Simpath”

1. Assign a full order to all edges from $e_1$ to $e_n$.
2. Constructing a binary decision tree by case-splitting on each edges from $e_1$ to $e_n$. 
Knuth’s algorithm “Simpath”

1. Assign a full order to all edges from $e_1$ to $e_n$.

2. Constructing a binary decision tree by case-splitting on each edges from $e_1$ to $e_n$. 

Not a s-t path

Not a simple s-t path
Knuth’s algorithm Simpath

6 connecting to s
5 connecting to 7
“simpath” for US map in Knuth-book
Path enumeration over Japan

Trip all prefectures from Hokkaido to Kagoshima by ground transportation.

<table>
<thead>
<tr>
<th></th>
<th>Vertices</th>
<th>Time (sec)</th>
<th>BDD nodes</th>
<th># of Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Up to once</td>
<td>47</td>
<td>0.01</td>
<td>951</td>
<td>$1.4 \times 10^{10}$</td>
</tr>
<tr>
<td>Up to twice</td>
<td>94</td>
<td>248.72</td>
<td>18,971,787</td>
<td>$5.0 \times 10^{44}$</td>
</tr>
</tbody>
</table>

14,797,272,518 ways

5,039,760,385,115,189,594,214,594,926,092,397,238,616,064 ways

(= 503正9760潤3851溝1518穏9594杼2145垓9492京6092兆3972億3861万6064)
Comparison with conventional ZDD generation

- **Conventional method:** Repeating logic/set operations between two ZDDs.
  - Based on Bryant’s “Apply” algorithm

- **Frontier-based method:** Direct ZDD generation by traversing a given graph.
  - Dynamic programming using a specific problem property.
Frontier-based method (generalization of simpath)

- Variation of s-t path problem
  - s-t paths $\rightarrow$ Hamilton paths (exercise in Knuth-book)
  - paths $\rightarrow$ cycles (also in Knuth-book)
  - Non-directed graphs $\rightarrow$ directed graphs
  - $\rightarrow$ Multiple s-t pairs (non crossing routing problem)

- Other various graph enumeration problems
  - Subtrees / spanning trees, forests, cutsets, k-partitions, connection probability, (perfect) matching, etc.

- Generating BDDs for Tutte polynomials (graph invariant)
  - We found that Sekine-Imai’s idea in 1995 was in principle similar to Knuth simpath algorithm.
  - They used BDDs instead of ZDDs.
  - Enumerating connective subgraphs, not paths.

\[
T(x, y) = \sum_{A \subseteq E} (x - 1)^{\rho(E)} - \rho(A) (y - 1)^{|A|} - \rho(A)
\]
Prof. Hayashi at Waseda Univ. (ERATO collaborator)

A leader of smart grid technology in power electric community. He receives much more attention after the earthquake.

Control of electricity distribution networks are so important after the nuclear plant accident, since solar and wind power generators are not stable.

We truly want to contribute something to the society as leading researchers of information technology.

→ We accelerate our collaborative work after the earthquake.
Switching power supply networks:

- Each district must connect to a power source (no black out).
- Two power sources must not directly connect.
- Too much currency may burn a line.
- Too long line may cause voltage shortage.

Huge number of patterns.

This trivial example has 14 switches. There are 210 feasible patterns out of 16384 combinations.

A typical real-life NW has 468 switches. We have to search the patterns out of $10^{140}$ combinations.
Application to power supply networks

- Graph $k$ partition problem ($k$-cut set enumeration)
  - Enumerate all partitions s.t. given $k$-vertices not together.
    - Every area is supplied from one power feeder.
  - Frontier-based method is effective.

- We succeeded in generating a ZDD of all solutions for a realistic benchmark with 468 control switches.
  - ZDD nodes: 1.1 million nodes (779MB), CPU time: ~20 min.
  - Number of solutions: $10^{63}$
    (213682013834853291168261221480495609817839244385235398189521540)
Applications of frontier-based method

Related to various real-life important problems

- Path enumeration for directed/undirected graphs
  - GIS (car navigation, railway navigation)
  - Dependency/Fault analysis industrial systems
  - Solving puzzles (Numberlink, Slitherlink, etc.)
  - Enumerating all possible concatenations of substrings

- Graph k-partition problems
  - Control of electric power distribution networks
  - Layout of refugee shelters for earthquake and tsunami
  - Design of electoral districts for democratic fairness

- Such civil engineering system networks are usually near to grid or planar graphs.

→ Frontier-based method is effective in many cases.
Advantages of generating ZDDs

- Not only enumeration but also giving an index structure.
- Not only indexing but also providing rich operations.
- Well-compressed structure for many practical cases.

- Related to various real-life important problems.
  - GIS (car navigation, railway navigation)
  - Dependency/Fault analysis industrial systems
  - Solving puzzles (Numberlink, Slitherlink, etc.)
  -Enumerating all possible concatenations of substrings
  - Control of electric power distribution networks
  - Layout of refugee shelters for earthquake and tsunami
  - Design of electoral districts for democratic fairness
Open software: “Graphillion.org”

- Toolbox for ZDD-based graph enumeration.
- Easy interface using Python graph library.
Tutorial video for “Graphillion”

もし、おねえさんが Graphillion を知っていたら…
If she'd known Graphillion at that time…

Graphillion: 数え上げおねえさんを救え / Don't count naively

JST Channel

17,507

1,913
Recent result on “power of enumeration”

- We proposed a novel p-value correction procedure “LAMP.” [PNAS: Terada et al. 2013]
  - Accurate statistical assessment to discover combinatorial factors.
  - Based on frequent itemset enumeration techniques.
- Our result shows that: state-of-the-art enumeration techniques can contribute to many kinds of experimental sciences.

Statistical significance of combinatorial regulations

Aika Terada\textsuperscript{a,b,c}, Mariko Okada-Hatakeyama\textsuperscript{d}, Koji Tsuda\textsuperscript{c,e,1}, and Jun Sese\textsuperscript{a,b,1}

\textsuperscript{a}Department of Computer Science and \textsuperscript{b}Education Academy of Computational Life Sciences, Tokyo Institute of Technology, Meguro-ku, Tokyo 152-8550, Japan; \textsuperscript{c}Minato Discrete Structure Manipulation System Project, Exploratory Research for Advanced Technology, Japan Science and Technology Agency, Sapporo, Hokkaido 060-0814, Japan; \textsuperscript{d}Laboratory for Integrated Cellular Systems, RIKEN Center for Integrated Medical Sciences (IMS-RCAI), Yokohama, Kanagawa 230-0045, Japan; and \textsuperscript{e}Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology, Koto-ku, Tokyo 135-0064, Japan

Edited by Wing Hung Wong, Stanford University, Stanford, CA, and approved July 3, 2013 (received for review February 4, 2013)

More than three transcription factors often work together to enable cells to respond to various signals. The detection of combinatorial regulation by multiple transcription factors, however, is not only deliberately excluding such tests. Here, we propose an efficient branch-and-bound algorithm, called the “limitless arity multiple-testing procedure” (LAMP). LAMP counts the exact number of
Summary

- Focus on “discrete structure manipulation system.”
  - Fundamentals for various practical applications.

- Based on BDDs/ZDDs.
  - Representing “logic” and “set,” primitive models of discrete structures.
  - We will consider higher-level algebraic structures.

- Recent results
  - Demonstration video: 1.5 million views!
  - Now we have a world record on this problem.
  - State-of-the-art enumeration has many practical applications.
    - power distribution network, railways, water/gas supply, etc.

- Let’s enjoy our discussion in this seminar.