<table>
<thead>
<tr>
<th>項目</th>
<th>内容</th>
</tr>
</thead>
<tbody>
<tr>
<td>タイトル</td>
<td>πDD: Permutation Decision Diagram based on Permutation Family Algebra</td>
</tr>
<tr>
<td>著者</td>
<td>Minato, Shin-ichi</td>
</tr>
<tr>
<td>発表日</td>
<td>2011-06</td>
</tr>
<tr>
<td>項目</td>
<td>Doc URL: <a href="http://hdl.handle.net/2115/48359">http://hdl.handle.net/2115/48359</a></td>
</tr>
<tr>
<td>項目</td>
<td>Type: conference presentation</td>
</tr>
<tr>
<td>項目</td>
<td>Note: ERATO湊離散構造処理系プロジェクト: 2010年度初冬のワークショップ（札幌北広島クラッセホテル）11月29日(月)～12月1日(水).</td>
</tr>
<tr>
<td>備考</td>
<td>未定</td>
</tr>
</tbody>
</table>
Direction of our research

Current
ZDDs

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

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

Still many applications remains where ZDDs would be effective.

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

Develop special new algebraic operations.

Family of permutations

Family of sets:
- Don’t consider order and duplication of items
- “abcc” and “bca” are the same.

Family of sequences:
- Distinguishes all finite sequences.
- \( \varphi, \{ \}, \{ \text{ab}, \text{aba}, \text{bbc} \}, \{ \text{a}, \text{aa}, \text{aaa}, \text{aaaa} \}, \text{etc.} \)

Family of permutations:
- Set of orders in a fixed number of items.
- \( \varphi, \{ 123 \}, \{ 12, 21 \}, \{ 123456, 132456, 246135 \} \)

Applications

Rubik’s cube: Let \( P = \{ \pi \} \) any primitive move of cube.)
- \( P \) includes 12 (= 2 ways \( \times \) 6 faces) permutations.
- Cartesian product \( P \times P \) represents all possible patterns obtained by twice of primitive moves.
- \( P^n \) will have all possible patterns. (but maybe too large.)

15 puzzle, Tower of Hanoi
- Optimization of packing / arranging strategy
- “Amida-drawing” (rudder-style swapping graph)
- One-to-one matching problems between two parties.
- A permutation corresponds to a bijective relation.
- Design of loss-less codes.
- Analysis of reversible logic. (related to quantum logic circuit.)

Permutations

Notation of permutation is often confusing.
(ex.) \( \pi = “246135” \quad (\neq “415263”) \)

\[
\begin{array}{ccc}
\pi & x & x^2 \\
1 & 1 & 1 \\
2 & 2 & 2 \\
3 & 3 & 3 \\
4 & 4 & 4 \\
5 & 5 & 5 \\
6 & 6 & 6 \\
\end{array}
\]

\[
\begin{array}{cc}
x & x^2 \\
1 & 1 \\
2 & 2 \\
3 & 3 \\
5 & 5 \\
6 & 6 \\
\end{array}
\]

\[
\begin{array}{cc}
x & x^2 \\
1 & 1 \\
2 & 2 \\
3 & 3 \\
5 & 5 \\
6 & 6 \\
\end{array}
\]
**Required properties for πDDs**

- Empty set \( \varnothing \) should be 0-terminal node.
- Singleton set of the identical permutation: \{ “123456789…” \} should be 1-terminal node.
  - We may write \{ e \} since we don’t have to consider the dimension (number of items) for the identical relation.
- \{ “132”, “321” \} and \{ “132456789”, “321456789” \} had better be represented in a same DD.
  - “Dimension of permutation” \( \dim(\pi) \) is defined as the largest ID relevant to the permutation. (We put \( \dim(e) = 0 \).)
  - “Dimension of family of permutations” \( \dim(P) \) is the largest dimension of permutation in the family. (We put \( \dim(\varnothing) = 0 \).)
- \( \pi \rightarrow \dim(P) \) should be the top-ID of πDD for \( P \).

**Decomposition of permutation by \( \tau_{xy} \)**

\[ \pi = "35214" \]

\[ \tau_{xy} \text{ (transposition)} \]

\[ \begin{array}{c}
\tau_{32} \\
\tau_{41} \\
\tau_{41} \\
\end{array} \]

\[ \begin{array}{c}
5 \\
5 \\
5 \\
\end{array} \]

\[ \begin{array}{c}
4 \\
4 \\
4 \\
\end{array} \]

\[ \begin{array}{c}
2 \\
2 \\
2 \\
\end{array} \]

\[ \begin{array}{c}
1 \\
1 \\
1 \\
\end{array} \]

\( \pi = "35214" = e \tau_{32} \tau_{41} \tau_{41} \tau_{54} \)

\( \pi \rightarrow \) canonical form

**Main idea of πDDs**

- Using a pair of IDs for each decision node.
  - Let \( x \) as \( \dim(P) \), and \( x > y > 0 \)
  - \( P = P_0 \cup P_1 \tau_{xy} \)
  - \( P_0 = \{ \pi \in P \mid x \pi \neq y \} \)
  - \( P_1 = \{ \pi \in (P \tau_{xy}) \mid x \pi = x \} \)
  - \( \dim(P_0) \leq \dim(P) \)
  - \( \dim(P_1) < \dim(P) \)

**Rule of variable ordering in πDDs**

- General rule \( x > y > 0 \)
- Rules for 0-edge side
  - if \( (x \geq x_0, y < y_0) \)
- Rule for 1-edge side \( x > x_1 \)

**Node reduction rules for πDDs**

- Same reduction rules as ZDDs.
  - Ordinary BDD rules don’t work.


**πDDs of single permutation**

\[ \begin{align*}
\varnothing & \{e\} \\
2 & 1 \\
\{21\} & \{132\} & \{312\} & \{321\} & \{231\}
\end{align*} \]

\[ \begin{align*}
0 & 1 & 2 & 3 & 4 & 5 \\
0 & 1 & 2 & 3 & 4 & 5 \\
\end{align*} \]

**πDDs for sets of permutations**

\[ \begin{align*}
\{21\} & \{213,312,321,231\} & \{e,213,132,321,231\} \\
\{213,312\} & \{e,213,132,312\} \\
\{21\} & \{e,21\} \\
\{21\} & \{e,21\} \\
\end{align*} \]

**Related work in Knuth-book**

*Generating all permutations*.

(Vol. 4. Fascicle 2)

**πDDs of single permutation**

\[ \begin{align*}
(35214) & \\
\{5,4\} & \{2,1\} \\
\{5\} & \{\varnothing\} \\
\end{align*} \]

**Related work in Knuth-book (cont.)**

```
7.2.1.2. Generating all permutations.
```

**Algebraic operations for πDDs**

- **"Permutation family algebra"**
  - \( \psi \{ e \} \): Empty and identical permutation. (0/1-terminal)
  - \( P; \varphi \): Returns the dimension of \( P \). (Item ID \( \varphi \) of the root node)
  - \( P; \rho \): Returns the largest ID with \( P; \varphi \). (Item ID \( \rho \) of the root node)
  - \( P; \text{factor}(x, y) \): Returns \( \{ \varnothing \in P \mid x = x \} \)
  - \( P; \text{top}(x) \): Returns \( P; \varphi \).
  - \( P; \text{top}(x) \): Returns union, intersection, and difference set.
  - \( P; \text{count} \): Counts number of combinations in \( P \).
  - \( P \times Q \): Cartesian product set of \( P \) and \( Q \).
  - \( P \div Q \): Quotient set of \( P \) divided by \( Q \). (Right-side division)
  - \( P \div Q \): Remainder set of \( P \) divided by \( Q \). (Right-side division)

\( \varnothing \) and \( \{ e \} \) have a strong relationship with Knuth’s tree structure for generating all permutations.

Nov 29, 2010 Shin-ichi Minato 15

Nov 29, 2010 Shin-ichi Minato 16

Nov 29, 2010 Shin-ichi Minato 17

Nov 29, 2010 Shin-ichi Minato 18
Synthesis of $\pi$DDs by algebraic operations

- $P \times Q = \{ (p,q) | \forall p \in P, \forall q \in Q \}$.
- Not independent of $\tau_y$ operation.

$P \times Q = (P_0 \cup P_1 \tau_{xy}) \times (Q_0 \cup Q_1 \tau_{xy})$

$= (P_0 \times Q_0) \cup (P_0 \times Q_1 \tau_{xy})$

$\cup (P_1 \times Q_0) \cup (P_1 \times Q_1 \tau_{xy})$

$= (P_0 \times Q_0) \cup (P_0 \times Q_1 \tau_{xy})$

$\cup (P_1 \times Q_0) \cup (P_1 \times Q_1 \tau_{xy})$

$\tau_{xy}$ operation

Let $\pi = (35214) = \tau_{12} \tau_{32} \tau_{41} \tau_{54}$

$\tau_{xy}$ union

Swap of cascaded $\tau_{xy} \tau_{uv}$

If $x > u > y = v$

$y' = u$

$u' = u$

$y' = v$

We can keep canonical form: $u' < x$

Rules to swap $\tau_{xy} \tau_{uv}$ to $\tau_{u'y} \tau_{xy}'$

- if $(u < v)$ (consider $\tau_{r_{u'y}}$)
  - if $(x < u$ or $u = v)$ (no swap needed)
  - if $(x > u > y = v)$
    - $y' = u$
    - $u' = u$
  - if $(x = u > y > v)$
    - $y' = u$
    - $u' = v$
  - if $(x = u > y > v)$
    - $y' = u$
    - $u' = y$

Otherwise:

$y' = y$

We can keep canonical form: $u' < x$
Algorithm of \((P \tau_{uv})\)

\[ P \tau_{uv} = \begin{cases} P_0 & \text{if } u = v \\ P_1 \tau_{uv} & \text{if } u < v \end{cases} \]

\[ P \tau_{uv} = (P_0 \bigcup P_1 \tau_{uv}) \tau_{uv} \]

if \((x \leq u)\)

\[ P \tau_{uv} = P_0 \tau_{uv} \bigcup P_1 (\tau_{xy} \tau_{uv}) = (P_0 \tau_{uv}) \bigcup (P_1 \tau_{uv}) \tau_{xy} \]

Recursive calls with cache.

Cartesian product operation

\[ P * Q = \{ (p, q) \mid p \in P, q \in Q \} \]

Now we got a recursive algorithm using operation \((P \tau_{uv})\).

\[ P * Q = (P * Q_0) \bigcup (P * Q_1) \tau_{xy} \]

Recursive calls with cache.

Product operation for disjoint permutations

\[ \{3,2\} \times \{2,1\} \rightarrow \{3,2,1\} \]

Product operation seems difficult.

- Less nodes shared.
- Product operation seems difficult.

Procedure of Factor operation

\[ P.\text{factor}(u, v) = (P_0 \bigcup P_1 \tau_{uv}).\text{factor}(u, v) \]

if \((x < u \text{ or } x < v)\)

\[ P.\text{factor}(u, v) = P \tau_{uv} \]

if \((x = u \text{ or } v)\)

\[ P.\text{factor}(u, v) = P_0.\text{factor}(u, v) \]

if \((x < u \text{ and } x < v)\)

\[ P.\text{factor}(u, v) = P_1.\text{factor}(u, v) \]

if \((x = u \text{ and } x = v)\)

\[ P.\text{factor}(u, v) = P \]

If we use SeqBDDs for permutations

- Less nodes shared.
- Product operation seems difficult.
πDDs for sets of permutations

\{21\}
\begin{align*}
\pi & \rightarrow 2 \rightarrow 1 \rightarrow e \\
& \left\{21, 312, 321, 231\right\} \\
& \left\{e, 213, 132, 312, 231\right\}
\end{align*}
\begin{align*}
\pi & \rightarrow 3 \rightarrow 3 \rightarrow 1 \rightarrow e \\
& \left\{21, 312, 321\right\} \\
& \left\{e, 213, 132\right\}
\end{align*}
\begin{align*}
\pi & \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow e \\
& \left\{21, 312\right\} \\
& \left\{21, 3, 2\right\}
\end{align*}
\begin{align*}
\pi & \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow e \\
& \left\{3, 2, 1\right\} \\
& \left\{3, 2, 1\right\}
\end{align*}

Upper bound of πDD sizes

- Number of Families of permutations up to \(n\) items: \(2^n!\)
  - \(n!\): 1, 1, 2, 6, 24, 120, …
  - \(2^n!\): 2, 2, 4, 64, 16777216, 1329227995784915872903807060280344576, …
- At least \(\log n\) bit needed to distinguish \(n\) objects.
  - Thus, πDD size can be \(O(n!)\) bit.

πDD sizes for typical cases

- \(\phi, \{e\}\): \(O(1)\) nodes
- Sets of a single permutation with \(n\) items: \(O(n)\) nodes
- Sets of any \(k\) permutations with \(n\) items: \(O(kn)\) nodes
- Sets of all \(n\) rotations with \(n\) items: \(O(n^2)\) nodes
- Sets of all \(n!\) permutations with \(n\) items: \(O(n^2)\) nodes

- Nodes for each permutation is bounded by “swap distance” from identical permutation.
- πDD can be compact for representing the family consists of many similar sub-permutations.

TODO

- Implementation of the algorithms.
- Determine complexity of operations.
- Applying to interesting problems.
  - Performance evaluation.
- Variable ordering problem.
- Relationship to permutation group theory.
  - If \(P * P = P\), then \(P\) forms a permutation group.
- Variations.
  - Histogram (multiset) of permutations.
  - Permutations of \(k\) out of \(n\) items. (allows lack of items)
  - Permutations of multiset items. (allows duplication of items)