<table>
<thead>
<tr>
<th>項目</th>
<th>内容</th>
</tr>
</thead>
<tbody>
<tr>
<td>項目</td>
<td>内容</td>
</tr>
<tr>
<td>項目</td>
<td>内容</td>
</tr>
<tr>
<td>項目</td>
<td>内容</td>
</tr>
<tr>
<td>項目</td>
<td>内容</td>
</tr>
</tbody>
</table>

※このファイルは、 Hokkaido University Collection of Scholarly and Academic Papers から提供されたものです。
**Direction of our research**

Current ZDDs
- Data mining, Machine learning
- Advanced searching etc.

(Combinatorial)

Further outputs
- Advanced ZDD-like structure
- Sequence data analysis
- Numerical data processing
- Processing of trees or semi-structured data

(Algebraic)

Applications with higher data model
- Develop special new algebraic operations.

Still many applications remains where ZDDs would be effective.

**Background**

- BDD: Boolean function
  - Boolean algebra
- ZDD: Family of sets
  - Family algebra
- ZDD-Vector: Histogram of itemsets
  - Itemset histogram algebra
- Sequence BDD: Family of sequences
  - Sequence family algebra

πDD: Family of permutations
- Permutation family algebra

**Applications**

- Rubik’s cube: Let $P = \{ \pi \mid \text{any primitive move of cube.} \}$
  - $P$ includes 12 (= 2 ways $\times$ 6 faces) permutations.
  - Cartesian product $P \times P$ represents all possible patterns obtained by twice of primitive moves.
  - $P^{16}$ 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.)

**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.
  - $\varphi, \{\}, \{ab, aba, bbc\}, \{a, aa, aaa, aaaa\}$, etc.
- Family of permutations:
  - Set of orders in a fixed number of items.
  - $\varphi, \{123\}, \{12, 21\}, \{123456, 132456, 246135\}$

**Permutations**

- Notation of permutation is often confusing.

(ex.) $\pi = “246135”$ (≠ “415263”)

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

$\begin{array}{ccc}
x & y = x \pi & z = y \pi = x \pi^2 \\
5 & 6 & 3 \\
3 & 4 & 1 \\
6 & 4 & 5 \\
5 & 3 & 2
\end{array}$
### Required properties for \( \pi \text{DDs} \)

- Empty set \( \emptyset \) 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” \( \text{Dim}(\pi) \) is defined as the largest ID relevant to the permutation. (We put \( \text{Dim}(e) = 0 \).)
- “Dimension of family of permutations” \( \text{Dim}(P) \) is the largest dimension of permutation in the family. (We put \( \text{Dim}(\emptyset) = 0 \).)
  \( \text{Dim}(\pi) \) should be the top-ID of \( \pi \text{DD} \) for \( P \).
  \( \text{Dim}(P) \) should be the top-ID of \( \pi \text{DD} \) for \( P \).

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

\[ \pi = "35214" \]

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

\[ \begin{cases} x \rightarrow y \\ y \rightarrow x \end{cases} \]

\[ \pi \tau_{xy} \text{ where } x\pi = y \]

\[ \begin{cases} z \rightarrow x \\ x \rightarrow z \end{cases} \]

\[ \begin{cases} y \rightarrow x \\ x \rightarrow y \end{cases} \]

### Required properties for \( \pi \text{DDs (cont.)} \)

- Each path from root node to 1-terminal node should correspond to a permutation in \( P \).
- Number of paths equals the cardinality of \( P \).
- Giving of canonical (unique) representation for a family of permutations.
- Efficient equivalence checking
- ZDD-like algebraic operations over \( \pi \text{DDs} \).
- Computation time depends on \( \pi \text{DD} \) size, not directly depend on cardinality of \( P \).

### Main idea of \( \pi \text{DDs} \)

- Using a pair of IDs for each decision node.

Let \( x \) as \( \text{Dim}(P) \), and \( x > y > 0 \)

\[ P = P_0 \cup P_1 \tau_{xy} \]

\[ P_0 = \{ \pi \in P \mid x \pi \neq y \} \]

\[ P_1 = \{ \pi \in (P \tau_{xy}) \mid x \pi = x \} \]

\[ \text{Dim}(P_0) \leq \text{Dim}(P) \]

\[ \text{Dim}(P_1) < \text{Dim}(P) \]

### Rule of variable ordering in \( \pi \text{DDs} \)

- General rule
  \( x > y > 0 \)

- Rules for 0-edge side
  \( x \geq x_0 \)
  if \( (x = x_0), y < y_0 \)

- Rule for 1-edge side
  \( x > x_1 \)

### Node reduction rules for \( \pi \text{DDs} \)

- Same reduction rules as ZDDs.
  - Ordinary BDD rules don’t work.
$\pi$DDs of single permutation

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>{e}</td>
<td></td>
</tr>
</tbody>
</table>

$\pi$DDs for sets of permutations

- {21}
- {213,312,321,231}
- {e,213,132,312,231}
- {e,213,132,312}
- {e,21,231}
- {21,231}

$\pi$DD has a strong relationship with Knuth's tree structure for generating all permutations.

Related work in Knuth-book

7.2.1.2. Generating all permutations.

"Inversion table."

Each permutation can be represented as combinations.

Algebraic operations for $\pi$DDs

- "Permutation family algebra"

<table>
<thead>
<tr>
<th>$\psi$, {e}</th>
<th>Empty and identical permutation. (0/1-terminal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>Returns the dimension of $P$. (Item ID $x$ of the root node)</td>
</tr>
<tr>
<td>$P$</td>
<td>Returns the largest ID with $P$ (item ID $y$ of the root node)</td>
</tr>
<tr>
<td>$P$</td>
<td>Returns $x \in P$ if $x \in y$</td>
</tr>
<tr>
<td>$P$</td>
<td>Returns $P \cap y$</td>
</tr>
<tr>
<td>$P$</td>
<td>Returns union, intersection, and difference set.</td>
</tr>
<tr>
<td>$P$</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 $Q$. (Right-side division)</td>
</tr>
<tr>
<td>$P % Q$</td>
<td>Remainder set of $P$ divided by $Q$. (Right-side division)</td>
</tr>
</tbody>
</table>
Synthesis of $\pi$DDs by algebraic operations

Cartesian product operation

Binary set operations between $\pi$DDs

Swapping of cascaded $\tau_{xy}$ and $\tau_{uv}$

Rules to swap $\tau_{xy}$ and $\tau_{uv}$ to $\tau_{uv'}\tau_{xy}$

If $x > u$ and $y = v$,

- $y' = u$
- $u' = u$
- $v' = y$

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

if $(u=v)$ return $P$
if $(u<v)$ return $P \tau_{uv}$

$P \tau_{uv} = (P_0 \cup P_1 \tau_{xy}) \tau_{uv}$ if $(x<u)$

if $(x \geq u)$

$P \tau_{uv} = P_0 \tau_{uv} \cup P_1 \tau_{uv}$

$= (P_0 \tau_{uv}) \cup (P_1 \tau_{uv})$

Recursive calls with cache.

Cartesian product operation

- $P \times Q = \{p \times q \mid \forall p \in P, \forall q \in Q\}$

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

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

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

Recursive calls with cache.

Product operation for disjoint permutations

- Product operation seems difficult.
- Less nodes shared.

Procedure of Factor operation

$P \text{ factor}(u, v) = (P_0 \cup P_1 \tau_{xy}) \text{ factor}(u, v)$

if $(x<u)$ or $(x<v)$ $P \text{ factor}(u, v) = P$ (if $u = v$)
if $(x=u)$ or $(x=v)$ $P \text{ factor}(u, v) = P_0$
if $(x<u)$ and $(x<v)$ $P \text{ factor}(u, v) = P_1$

if $(x>v)$ or $(y>v)$ $P \text{ factor}(u, v) = P_0 \text{ factor}(u, v)$
if $(x<v)$ and $(y>v)$ $P \text{ factor}(u, v) = P_1 \text{ factor}(u, v)$
if $(x>v)$ and $(y<v)$ $P \text{ factor}(u, v) = P_0 \text{ factor}(u, v)$
if $(x>v)$ and $(y>v)$ $P \text{ factor}(u, v) = P_1 \text{ factor}(u, v)$

If we use SeqBDDs for permutations?

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

\{21\} \rightarrow 2 \rightarrow 1

\{213,312,321,231\} \rightarrow 3 \rightarrow 2 \rightarrow 1

\{e, 21\} \rightarrow 2 \rightarrow 1

\{e, 213,132,312,321,231\} \rightarrow 3 \rightarrow 2 \rightarrow 1

\{e, 213,132,312\} \rightarrow 3 \rightarrow 2

\{e,21\} \rightarrow 2 \rightarrow 1

\{e,213,132,312\} \rightarrow 3 \rightarrow 2

\{e,21\} \rightarrow 2 \rightarrow 1

\{213,312\} \rightarrow 3 \rightarrow 2 \rightarrow 1

\{e,21\} \rightarrow 2 \rightarrow 1

\{213,312,231\} \rightarrow 3 \rightarrow 2 \rightarrow 1

\{e,213,132,312,321,231\} \rightarrow 3 \rightarrow 2 \rightarrow 1

Upper bound of $\pi$DD 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, $\pi$DD size can be $O(n!)$ bit.

$\pi$DD sizes for typical cases

- $\emptyset, \{e\}$: $O(1)$ nodes
- Sets of a single permutation with $n$ items: $O(n)$ nodes
- Sets of any $k$ permutations with $n$ items: $O(k \cdot n)$ 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.
- $\pi$DD 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 $P \cdot P = P$, 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)