Graphillion: ZDD-Based Compilation Tool for Graph Enumeration and Random Sampling

Shin-ichi Minato
Kyoto University, Japan.
“ERATO” project and after

- **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.

**JSPS Basic Research Project:**
“Discrete Structure Manipulation System”
2015～2020, total 1M USD
Our recent activity: WEPA 2019

- The 3rd International Workshop on Enumeration Problems & Applications
  - Dagstuhl style WS with “Enumeration people”.
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**.

- design automation
- data mining / knowledge discovery
- fault analysis
- bio informatics
- machine learning / classification
- constraint satisfaction problem
- web data analysis

**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

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

2019.12.18 Shin-ichi Minato -Kocoon 2019
Effect of BDD reduction rules

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

$O(2^n)$

$O(n)$
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:**

\[
F = (a \ b \ \neg c) \lor (\neg b \ c)
\]

**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

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(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.
ZDD construction based on set operations

\[
\text{change}(c) \cup \text{union} \text{change}(b) \cup \text{union} \text{change}(a)
\]

\[
\{c\}, \{\lambda\}, 0, 1, \{b, bc\}, \{\lambda, b, bc, c\}
\]

\[
\text{change}(c) \cup \text{union} \text{change}(a)
\]

\[
\{\lambda\}, \{a\}, 0, 1, \{\lambda, a, b, bc, c\}
\]

\[
\text{change}(b) \cup \text{union}
\]

\[
\{\lambda, b, bc, c\}, \{\lambda, a, b, bc, c\}
\]
Example: ZDD for frequent itemsets

- The results of frequent itemsets are obtained as ZDDs on the main memory.

<table>
<thead>
<tr>
<th>Record ID</th>
<th>Tuple</th>
<th>Freq. thres. $\theta = 7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a\ b\ c$</td>
<td>${ ab, bc, a, b, c }$</td>
</tr>
<tr>
<td>2</td>
<td>$a\ b$</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$a\ b\ c$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$b\ c$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$a\ b$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$a\ b\ c$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$c$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$a\ b\ c$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$a\ b\ c$</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>$a\ b$</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>$b\ c$</td>
<td></td>
</tr>
</tbody>
</table>

"LCM-ZDD" algorithm
<table>
<thead>
<tr>
<th>Dataset name</th>
<th>min. support</th>
<th># solutions</th>
<th>LCM-ZDDs ZBDD</th>
<th>Time(s)</th>
<th>LCM-ZDDs LCM</th>
<th>Time(s)</th>
<th>Original LCM</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mushroom: 1,000</td>
<td>1,000</td>
<td>123,287</td>
<td>760</td>
<td>0.50</td>
<td>0.49</td>
<td>0.64</td>
<td>1.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>1,442,504</td>
<td>2,254</td>
<td>1.32</td>
<td>1.30</td>
<td>3.29</td>
<td>3.49</td>
<td></td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>5,259,786</td>
<td>4,412</td>
<td>2.25</td>
<td>2.22</td>
<td>9.96</td>
<td>5.11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>18,094,822</td>
<td>6,383</td>
<td>3.21</td>
<td>3.13</td>
<td>31.63</td>
<td>6.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>66,076,586</td>
<td>11,584</td>
<td>5.06</td>
<td>4.87</td>
<td>114.21</td>
<td>6.72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>153,336,056</td>
<td>14,307</td>
<td>7.16</td>
<td>7.08</td>
<td>277.15</td>
<td>6.97</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>198,169,866</td>
<td>17,830</td>
<td>8.17</td>
<td>7.86</td>
<td>357.27</td>
<td>6.39</td>
<td></td>
</tr>
<tr>
<td>T10l4D100K: 100</td>
<td>100</td>
<td>27,533</td>
<td>8,482</td>
<td>0.85</td>
<td>0.85</td>
<td>0.86</td>
<td>209.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>53,386</td>
<td>16,872</td>
<td>0.97</td>
<td>0.92</td>
<td>0.98</td>
<td>242.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>129,876</td>
<td>58,413</td>
<td>1.13</td>
<td>1.08</td>
<td>1.20</td>
<td>290.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>411,366</td>
<td>173,422</td>
<td>1.55</td>
<td>1.36</td>
<td>1.64</td>
<td>332.22</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1,923,260</td>
<td>628,491</td>
<td>2.86</td>
<td>2.08</td>
<td>3.54</td>
<td>370.54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>6,169,854</td>
<td>1,576,184</td>
<td>5.20</td>
<td>3.15</td>
<td>8.14</td>
<td>386.72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>19,561,715</td>
<td>3,270,977</td>
<td>9.68</td>
<td>5.09</td>
<td>22.66</td>
<td>384.60</td>
<td></td>
</tr>
<tr>
<td>BMS-WebView-1: 50</td>
<td>50</td>
<td>8,192</td>
<td>3,415</td>
<td>0.11</td>
<td>0.11</td>
<td>0.12</td>
<td>29.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>48,544</td>
<td>10,755</td>
<td>0.18</td>
<td>0.18</td>
<td>0.22</td>
<td>48.54</td>
<td></td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>461,522</td>
<td>28,964</td>
<td>0.49</td>
<td>0.42</td>
<td>0.98</td>
<td>67.16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>1,177,608</td>
<td>38,164</td>
<td>0.80</td>
<td>0.69</td>
<td>2.24</td>
<td>73.64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>4,849,466</td>
<td>49,377</td>
<td>1.30</td>
<td>1.07</td>
<td>8.58</td>
<td>83.36</td>
<td></td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>69,417,074</td>
<td>59,119</td>
<td>3.53</td>
<td>3.13</td>
<td>144.98</td>
<td>91.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1,531,980,298</td>
<td>71,574</td>
<td>31.90</td>
<td>29.73</td>
<td>3,843.06</td>
<td>92.47</td>
<td></td>
</tr>
<tr>
<td>chess: 1,000</td>
<td>1,000</td>
<td>29,442,849</td>
<td>53,338</td>
<td>197.58</td>
<td>197.10</td>
<td>248.18</td>
<td>1,500.78</td>
<td></td>
</tr>
<tr>
<td>connect: 40,000</td>
<td>40,000</td>
<td>23,981,184</td>
<td>3,067</td>
<td>5.42</td>
<td>5.40</td>
<td>49.21</td>
<td>212.84</td>
<td></td>
</tr>
<tr>
<td>pumsb: 32,000</td>
<td>32,000</td>
<td>7,733,322</td>
<td>5,443</td>
<td>60.65</td>
<td>60.42</td>
<td>75.29</td>
<td>4,189.09</td>
<td></td>
</tr>
<tr>
<td>BMS-WebView-2: 5</td>
<td>5</td>
<td>26,946,004</td>
<td>353,091</td>
<td>4.84</td>
<td>3.62</td>
<td>51.28</td>
<td>118.07</td>
<td></td>
</tr>
</tbody>
</table>
We can extract distinctive itemsets by comparing frequent itemsets for multiple sets of databases.

Various ZDD algebraic operations can be used for the comparison of the huge number of frequent itemsets.
Post Processing after ZDD construction

- We can extract distinctive itemsets by comparing frequent itemsets for multiple sets of databases.
- Various ZDD algebraic operations can be used for the comparison of the huge number of frequent itemsets.

Knowledge compilation into ZDDs
Graph enumeration using BDDs/ZDDs

- Enumerating subgraphs satisfying a constraints for a given graph.
  - A variable for each edge (use or not).
  - A path in ZDD corresponds to a subgraph.
  - SAT: 1-terminal, UNSAT: 0-terminal.

- For Boolean expression of constraints, a set of subgraph can be a BDD/ZDD.
  - Well-compressed if there are many similar subgraphs.
  - ZDD is better than BDD if using an edge disables another one.
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 $\perp$.) 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 $P_8 \boxtimes P_8$, 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 $P_6 \boxtimes P_6$ has exactly 603,841,648,931 simple cycles.
26 x 26: Our record in Nov. 2013
(1404 edges included in the graph.)

- up to 18 × 18, we can construct a ZDD to keep all solutions.
- from 19 × 19, we just count the number of solutions.
- from 22 × 22, we only consider the n × n grid graphs.
From upper-ordered edges, by case-splitting by using or not, a binary decision tree is constructed.

- Terminate to “0” if constraint-violation found during the process.
- Terminate to “1” if constraint-satisfied after all decisions.
From upper-ordered edges, by case-splitting by using or not, a binary decision tree is constructed.

- Terminate to “0” if constraint-violation found during the process.
- Terminate to “1” if constraint-satisfied after all decisions.

Equivalent nodes should be shared. Redundant operations can be avoided.
"Frontier" for ZDD node sharing

6 connecting to 1
5 connecting to 7

Frontier

vertex | 1 2 3 4 5 6 7 8 9
mate[v] | 6 0 0 0 7 1 5 8 9

Meaningful information

2019.12.18
**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.

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

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

- Frontier-based method:
  - Direct ZDD construction by top-down & breadth-first.
    - Enumerating paths/cycles, trees/forests, connected parts, etc.
    - Dynamic programming using a specific problem property.
      (path-width)
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
Summary: KC tool with 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 refuge shelters for earthquake and tsunami
  - Design of electoral districts for democratic fairness