Finding good variable ordering for efficient evaluation of network reliability with BDDs

Tuomas Tirronen
Department of Communications and Networking
Aalto University

Outline

1. The reliability calculation problem
2. Reliability using binary decision diagrams (BDDs)
3. Variable ordering problem
4. Optimal orderings for small cases
5. Heuristics / examples
Reliability of communication networks

• **Approach:** reliability evaluation of a network graph where edges have nonzero failure probability
  - reliability = probability of connectivity for some set K of network nodes
  - performability etc are not taken into account at this point
  - idealistic model – not taking account layers, protocols

• **Original motivation:** finding efficient way to calculate the exact reliability of realistic sized networks, possibly taking node failures into account as well

Exact evaluation of reliability

Scenarios:
1. Edge failures / node failures
2. Independent / dependent failures
3. Size of set K (2-terminal, K-terminal, all-terminal reliability)

• The problem is NP-hard for general networks
• Typical approaches:
  1. Use recursion and decomposition (factoring) of network graph to generate exact symbolic expression for reliability
  2. Enumerate (minimal) cuts/paths in the network to generate the structure function
**Inefficiency of reliability calculation algorithms**

1. Typically complexity grows exponentially with network size
2. Isomorphic sub-problems are solved separately
3. Large memory/storage space required to store results
4. No efficient way to take into account additional information (such as imperfect nodes)


---

**Reliability calculation using Binary Decision Diagrams**

- **Motivation:**
  1. Efficient data structure for storing complex boolean functions (thus also reliability expressions)
  2. Isomorphic sub-graphs can be reused when doing network decomposition
  3. Essential variables can be found efficiently
  4. Subsequent modifications of the BDD can be done efficiently
     - e.g., taking node failures into account → incident edge substitution
Binary decision diagrams (BDDs)

- BDDs are data structures for Boolean expressions
  - directed, acyclic graph
  - decision nodes with two descendants each and two terminal nodes
  - Variable ordering determines the size of the BDD
    - the order in which we evaluate each of the variables
  - reduced ordered BDDs \( \rightarrow \) compact and canonical form of a Boolean expression

From full binary decision tree to BDD

\[ x_1 \land x_2 \lor x_3 \]
Using BDDs for reliability evaluation

- **Basic idea:** Boolean expression stores the information of all possible paths in the network
- The BDDs are generated with recursive decomposition taking into account possible isomorphisms in subgraphs
- **Problem:** finding a good variable ordering for the network

Variable ordering

- Fixing the ordering makes the (reduced) BDD the *canonical form* of a boolean expression
- Also determines the **size of the BDD**
  - determines the efficiency of reliability calculation and other manipulations
  - example: carry-out adder, best $O(n)$, worst $O(2^n)$
- Optimal ordering can be found
  - exhaustively: $O(n! \cdot 2^n)$
  - algorithm by Friedman and Supowit: $O(n^2 \cdot 3^n)$

Example: best and worst ordering

\[ e_1 > e_5 > e_4 > e_6 > e_3 > e_2 \]

Example: reliability calculation

\[ e_1 > e_5 > e_4 > e_6 > e_3 > e_2 \]

\[ 0.9 \cdot 0.9 + 0.9 = 0.99 \]

\[ 0.9 \cdot 0.9 \cdot 0.99 = 0.981 \]
Research questions

• Do the optimal orderings for small problems have some exploitable characteristics?
• What algorithms (heuristics) give good variable orderings?
  – also, what is a "good" ordering
• Are there some characteristics in (typical transport) networks which could be exploited when generating the ordering?
• Practicality: network sizes?

Optimal orderings for some small networks

• Optimal orderings were found using exhaustive search, BDDs were constructed for all
  – Implemented in Mathematica
  – maximum size: ~6 nodes ~10 edges
• Examples
Example: Optimal ordering

Possible to swap 1. and 2., or 8. and 9.

"Heuristic": Start from source, put incident edges first, then go through the possible clique they form. Similarly handle sink node in the end of the order. Rest of the edges in the middle.

Heuristics: DFS and BFS

- Depth first and breadth first searches often give good orderings
- Problem: node (edge) order is somewhat random
  - implementation details affect
- Results in the literature are often vague
  - e.g. "we used DFS and found it was good enough"
- DFS seems to be more sensitive implementation-wise
Degree-order heuristic

- Use "edge degree" as a weight
  - the total degree of edge endpoints
- Pick the edge with the lowest weight, remove it from graph and recalculate weights
- Seems to work on par or little better than results reported in Kuo et al.
- Using this like the "heuristic" the optimal ordering problems suggests gives reasonably good results

---

Example

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Kuo et al.</th>
<th>BFS*</th>
<th>DFS*</th>
<th>Deg.ord.*</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>261</td>
<td>347</td>
<td>457</td>
<td>260</td>
</tr>
</tbody>
</table>

* implementation affects size
Heuristics

• From experimental point of view, good heuristics should give consistent results independent of implementation details
  – I have not found this to be true with the algorithms I have tested
• Improving the generated ordering
  – e.g. flipping two variables
  – NP-complete problem itself
  – I have not considered so far

Conclusion

• BDDs provide an alternative way to calculate exact network reliability
• However, variable ordering must be good otherwise not worth it
• The results reported in literature (for the reliability case) with different heuristics are not satisfactory
  – (In my opinion)
Discussion

Ordnung führt zu allen Tugenden. Was aber führt zur Ordnung?

Order leads to all virtues. But what leads to order?

- Georg Christoph Lichtenberg (1742-1799)