A note on independent-disjoint analysis of network reliability

R. E. Andersen
Department of Mathematics
Lafayette College
Easton, PA 18042 U.S.A.

A. O. Balan
Department of Computer Science
Brown University
Providence, RI 02912 U.S.A.

J. P. Rowe
Department of Computer Science
North Carolina State University
Raleigh, NC 27695 U.S.A.

L. Traldi (corresponding author)
Department of Mathematics
Lafayette College
Easton, PA 18042 U.S.A.
email traldil@lafayette.edu
telephone 610 330-5276; fax 610 330-5721

Abstract

We introduce the notion of independent-disjoint analyses of Boolean functions. These analyses include both SDPs and OBDDs, and are well suited for network reliability computations.

Keywords. Sum of disjoint products; Binary decision diagram; Network reliability; Boolean function

Network reliability calculations are both computationally intractable [9] and of great practical value; consequently many different network reliability algorithms have been developed. In particular, several different techniques of Boolean analysis have been used, including various kinds of sums of products (SOPs) and binary decision diagrams (BDDs). A Boolean reliability calculation in which every variable x has $\Pr(x) = 0.5$ is essentially a count of the number of satisfying truth assignments of the variables, i.e., an instance of the intractable problem SAT-COUNT. Most Boolean techniques were not introduced with reliability or SAT-COUNT calculations in mind, and many of these techniques are not well suited to such calculations – after a Boolean function is analyzed with one of these ill-suited techniques, reliability and SAT-COUNT computations remain intractable. For instance, an ordinary SOP is not well suited to reliability or SAT-COUNT; typically there are many operational states that satisfy more than one of the products in the SOP, and it is difficult to take this duplication into account when calculating the cumulative value. A classical solution is to analyze an SOP to yield a sum of disjoint products (SDP); see [10] for a survey of SDP algorithms used in network reliability. The intractability of reliability and SAT-COUNT is then reflected in the fact that for some types of Boolean functions, the SDPs are exponentially larger than the smallest SOPs. Similarly, an arbitrary BDD description of a Boolean function does not generally lead to a tractable reliability or SAT-COUNT computation (see [13] for a comprehensive discussion of BDDs). The computation can be performed conveniently if the BDD is converted into an ordered BDD (OBDD), but for some types of

Boolean functions the OBDDs are exponentially larger than the smallest BDDs.

We observe that for both BDDs and SOPs, the difference between being well- and ill-suited for computing reliability and SAT-COUNT can usually be explained very easily, using only the basic probabilistic ideas of disjointness and independence: an analysis of a Boolean function is well-suited to reliability calculations if it can be recursively expressed in terms of conjunctions and disjunctions of disjoint and independent Boolean functions. Two Boolean functions are *disjoint* if and only if there is no combination of truth values of the variables that satisfies both functions. *Independence* is more complicated, because it can happen that two Boolean functions represent independent events for some combinations of variable probabilities and not for others; but presuming that the variables are pairwise independent, A and B will always be independent if none of the variables that appear in A appear also in B.

The observation of the preceding paragraph suggests the following.

Definition. An independent-disjoint analysis (IDA) of a Boolean function is one of the following:

(a) a single variable or its negation, or

(b) $A + B$ or AB, where A and B are disjoint IDAs, or

(c) $A + B$ or AB, where A and B are independent IDAs.

A simple recursive calculation of the reliability of a Boolean function given in IDA form is based on four facts: if A and B are disjoint events whose probabilities are known then $\Pr(A \cup B) = \Pr(A) + \Pr(B)$ and $\Pr(A \cap B) = 0$, and if A and B are independent events whose probabilities are known then

$\Pr(A \cap B) = \Pr(A)\Pr(B)$ and $\Pr(A \cup B) = 1 - (1 - \Pr(A))(1 - \Pr(B))$. As this recursive calculation is so simple, the general intractability of reliability calculations implies that for some types of Boolean functions, all the IDAs are exponentially larger than a smallest possible description. Nevertheless a typical Boolean function has IDAs that are smaller than its SDPs and OBDDs. The reason is simple: the definition of IDAs is so general that it includes all SDPs and OBDDs along with many other forms.[1]

The generality of the definition implies that there are many different algorithms that could be used to produce IDAs. Here is an outline of one such algorithm, essentially a Boolean version of the factoring algorithm studied by Satyanarayana and Chang [11] for graphs and Barlow and Iyer [4] for more general systems, with a simplified element-choosing heuristic used in step 4. An implementation is freely available at http://ww2.lafayette.edu/∽traldil/idapage.html.

1. A Boolean reliability function with pairwise independent variables is input in SOP form. (A Boolean reliability function is given by an SOP which does not involve any negated variables; the products which appear in the smallest SOP for that function are its *minpaths*.)

2. Variables that are in series (i.e., no minpath includes some without the others) are grouped together, and each group is replaced by a single variable which represents the product of the grouped variables. Variables that are parallel (i.e., they are interchangeable and no minpath includes more than one)

[1] The observation that OBDDs are related to SDPs has certainly appeared elsewhere; for instance [14] describes an OBDD representation of a Boolean function is a "graph-based set of disjoint products" and [5] refers to an SDP as an "explicitly given list of products," clearly suggesting that an OBDD is an implicitly given list of disjoint products. However we do not know of any other structures that generalize both SDPs and OBDDs, as IDAs do.

are grouped together, and each group is replaced by a single variable which represents the sum of the grouped variables.

3. If step 2 has reduced the number of variables, it is applied again.

4. If there is more than one variable and step 2 does not reduce the number of variables, then a Shannon decomposition $f = xf_x + \bar{x}f_{\bar{x}}$ is performed, and the algorithm is then applied to f_x and $f_{\bar{x}}$ separately. (In our implementation we choose x to be one of the variables that appears in the largest number of minpaths.)

The output of this algorithm is naturally represented as a generalized BDD, with variable transformations at the nodes of the diagram representing the replacements implemented in step 2. The fact that variable transformations can be useful in reducing BDD size is well known; see for instance [2, 6, 7, 13]. The output of the algorithm might also be described as a generalized SDP, one that allows disjunctions of asserted variables and disjunctions of negated variables to appear in the products. SDPs that allow disjunctions of negated variables are commonly called MVI-SDPs (MVI stands for "multivariable inversion") but we have not seen SDPs that allow disjunctions of asserted variables mentioned in the literature.

To illustrate the small size of some IDAs, consider a well-known example of Abraham [1]. Here is the reliability function in SOP form, with 24 minpaths.

$$jkl + bcjl + acdh + dfhk + acfjl + bcdfh + abdhk + ghijk + efghk +$$
$$acegh + efikl + aceil + dehijk + abeghk + bcefgh + bcghij + abeikl +$$
$$bcefil + dfgikl + acdgil + acfghij + bcdehij + bcdfgil + abdgikl$$

SDP forms of this function have been discussed by many authors; see [3, 5, 8, 12] and their references. The function's smallest known ordinary SDP involves 53 disjoint products, and its smallest known MVI-SDP involves 35 disjoint products. The algorithm outlined above produces the much simpler IDA of the function given below, with only 11 disjoint summands. (N.b. Reducing the number of disjoint summands directly reduces the number of operations involved in evaluating the function, as no variable appears more than once in any summand.)

$$((e+f+a)d+l+g)ijhc(k+b)+\bar{\imath}jhc(k+b)(l+(d+eg)(f+a))+$$

$$((il+g)e+d)\bar{j}hc(k+b)(f+a)+(d+l+(i+e)g)\overline{(k+b)}hcajf+$$

$$\overline{(k+b)}hca\overline{(jf)}((il+g)e+d)+(f+ab)kh\bar{c}j(d+l+(i+e)g)+$$

$$(e(g+il)+d)(f+ab)kh\bar{c}\bar{j}+(i(de+g)+l)\overline{(f+ab)}jkh\bar{c}+$$

$$((a+b)c+k)(j+f)\bar{h}li(dg+e)+(kb+c)a\bar{h}li(dg+e)\overline{(j+f)}+$$

$$\overline{(i(dg+e))}j\bar{h}l((fa+b)c+k)$$

There is a similar simplification in BDD representations of the function. An application available at http://tech-www.informatik.uni-hamburg.de/applets/java-bdd/bdd-applet.html produces an OBDD with more than 70 internal nodes. As pictured below, a BDD with node transformations requires only 21 internal nodes to represent the IDA form of the function. In the figure each node contains an expression, and under each node the left-hand link represents the negation of that expression and the right-hand link represents the assertion of

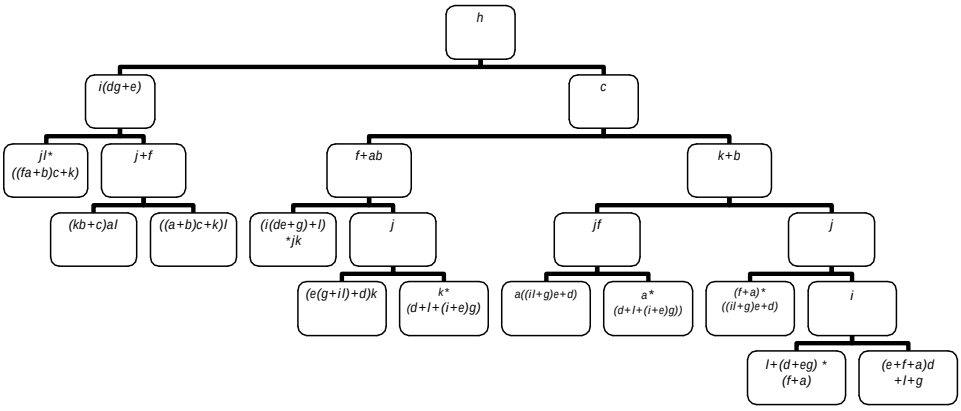

Figure 1: an IDA of Abraham's example

that expression. Technically, there should also be left-hand links to 0 and right-hand links to 1 from the nodes at the bottom of the diagram; these links have been omitted for simplicity.

In closing we observe that IDAs can be made even more flexible if we allow them to be combined using operations other than addition and multiplication. For instance, parts (b) and (c) of the definition might be modified to include $A \oplus B$, the exclusive disjunction, and $AB + AC + BC$, the 2-out-of-3 reliability problem. There are simple formulas for $\Pr(A \oplus B)$ and $\Pr(AB + AC + BC)$ when A, B and C are disjoint or independent, so evaluating the reliability of networks represented by the resulting modified IDAs will be convenient.

Acknowledgment

We are grateful to Lafayette College for its support of this work.

References

[1] J. A. Abraham, An improved method for network reliability, IEEE Trans Reliab R-28 (1979), 58-61.

[2] S. Aborhey, Binary decision tree test functions, IEEE Trans Comput 37 (1988), 1461-1465.

[3] A. O. Balan and L. Traldi, Preprocessing minpaths for sum of direct products, IEEE Trans Reliab 52 (2003), 289-295.

[4] R. E. Barlow and S. Iyer, Computational complexity of coherent sytems and the reliability polynomial, Prob Eng Inf Sci 2 (1988), 461-469.

[5] E. Châtelet, Y. Dutuit, A. Rauzy, and T. Bouhoufani, An optimized procedure to generate sums of disjoint products, Reliab Eng Syst Saf 65 (1999), 289-294.

[6] W. Günther and R. Drechsler,. Efficient minimization and manipulation of linearly transformed binary decision diagrams, IEEE Trans Comput 52 (2003), 1196-1209.

[7] Y. Jiang, S. Matic, and R. K. Brayton, Generalized cofactoring for logic function evaluation, Proc 40th Conf Design ACM, Anaheim, CA, 2003, pp. 155-158.

[8] M. O. Locks and J. M. Wilson, Nearly minimal disjoint forms of the Abraham reliability problem, Reliab Eng Syst Saf 46 (1994), 283-286.

[9] J. S. Provan and M. O. Ball, On the complexity of counting cuts and of computing the probability that a graph is connected, SIAM J. Comp. 12 (1983), 777-788.

[10] S. Rai, M. Veeraraghavan, and K. S. Trivedi, A survey of efficient reliability computation using disjoint products approach, Networks 25 (1995), 147-163.

[11] A. Satyanarayana and M. K. Chang, Network reliability and the factoring theorem, Networks 13 (1983), 107-120.

[12] L. Traldi, Non-minimal sums of disjoint products, Reliab Eng Syst Saf 91 (2006), 533-538.

[13] I. Wegener, Branching programs and binary decision diagrams, SIAM, Philadelphia, PA, 2000.

[14] F.-M.Yeh and S.-Y. Kuo, OBDD-based network reliability calculation, Elec Letters 33 (1997), 759-760.