Preprocessing MinPaths for Sum of Disjoint Products

Alexandru O. Balan and Lorenzo Traldi, *Member, IEEE*

***Abstract*—Network reliability algorithms which produce sums of disjoint products (SDP) are sensitive to the order in which the minimal pathsets are analyzed. The minpaths are *preprocessed* by choosing this order in the hope that an SDP algorithm will then provide a relatively efficient analysis. The most commonly used preprocessing strategy is to list the minpaths in order of increasing size. This paper gives examples for which this strategy is not optimal. A new preprocessing strategy which works well for SDP algorithms with single-variable inversion (SVI) is introduced. It is also observed that optimal preprocessing for SVI-SDP can be different from optimal preprocessing for SDP algorithms which use multiple-variable inversion; one reason for this is that MVI-SDP algorithms handle disjoint minpaths much more effectively than SVI-SDP algorithms do. Both kinds of SDP algorithms profit from prior reduction of elements and of subsystems which are in parallel or in series.**

***Index Terms*—Preprocessing, sum of disjoint products (SDP), system reliability.**

ACRONYMS[1]

i.i.d.	s-independent and identically distributed
MVI	multiple-variable inversion
SDP	sum of disjoint products
SVI	single-variable inversion
s	statistical(ly)

NOTATION

$a, b, c, \ldots$	variable representing the (operation of) i.i.d. elements of a system
$\bar{a}, \bar{b}, \bar{c}, \ldots$	variable representing the failure of an element
$P_1, P_2, \ldots$	variable representing the operation of all elements of a set
$\bar{P}_1, \bar{P}_2, \ldots$	variable representing the failure of at least 1 element of a set
$a + \bar{P}$	disjunction: a or $\bar{P}$
$a \cdot \bar{P}$	conjunction: a and $\bar{P}$
$\lvert P \rvert$	number of elements in set P
$P_i - P_j$	relative complements: $\{x \mid x \in P_i \text{ and } x \notin P_j\}$

I. INTRODUCTION

A coherent-binary-system on a finite set, E, is given by its minpaths, $P_1, \ldots, P_n$, which are subsets of E, and none of which contain any other. A (minpath-based) SDP algorithm calculates the system reliability by analyzing the disjoint events: P_1 (P_1 is operational), $P_2 \cdot \bar{P}_1$ (P_2 is operational, and P_1 fails), …, $P_n \cdot \bar{P}_{n-1}, \ldots, \bar{P}_1$ (P_n is operational and $P_1, \ldots, P_{n-1}$ all fail) separately, expressing each one as a union of disjoint events.

If $P_1, \ldots, P_n$ are listed in a different order, then the disjoint events $P_1, P_2 \cdot \bar{P}_1, \ldots, P_n \cdot \bar{P}_{n-1}, \ldots, \bar{P}_1$ are changed, and their expressions as disjoint sums change too. [1] suggests that to minimize the number of terms in the disjoint sums, smaller minpaths should be listed before larger ones. The reasoning behind the suggestion is simple and compelling: in general the smaller the relative complements $P_{j-1} - P_j, \ldots, P_1 - P_j$ are, the less complicated their intersections are, and the less difficult it is to analyze the event $P_j \cdot \bar{P}_{j-1}, \ldots, \bar{P}_1$. Listing smaller minpaths before larger ones tends to make these relative complements $P_i - P_j$ small, because P_i is small and P_j is large.

The preprocessing advice of [1] has been strongly endorsed in many papers, including [2]–[4], [6]–[9], [11], [13]. [6] conjectures that an SDP algorithm which uses the suggestion of [1] (together with other preprocessing for minpaths of the same size and an effective negation-simplifying technique) always produces an SDP analysis which is so nearly minimal that it is "probably not worth the effort" to refine the analysis further. [2] states baldly: "It is well known that for getting short disjoint sums the minimal path sets must be ordered in an increasing sequence according to the number of their elements." Some examples are discussed here, which indicate instead that the suggestion of [1], though it might be relatively effective in general, is often not optimal and sometimes quite far from optimal. This paper introduces a different preprocessing strategy which tends to follow the suggestion of [1] but is not absolutely bound to. The new strategy seems to do a reasonable job of, detecting situations in which the advice of [1] should be disregarded by SDP algorithms which do not involve multiple-variable inversion. Also mentioned here are some examples of systems for which all SDP algorithms result in highly nonoptimal analyses, regardless of preprocessing.

II. SOME EXAMPLES FOR SVI-SDP

The references, especially the survey article [9], show some of the many SDP algorithms which have been discussed in the literature. Differences among these algorithms include:

- the details of their preprocessing strategies (though all follow [1]'s suggestion of listing smaller minpaths first),
- details of the strategies they use to analyze the conjunctions of negations that appear in the disjoint events $P_j \cdot \bar{P}_{j-1}, \ldots, \bar{P}_1$,
- the fact that some SDP algorithms use multiple-variable inversion and some don't.

Manuscript received December 21, 2001; revised April 18, 2003. Responsible Editor: R. A. Evans.

The authors are with the Department of Mathematics, Lafayette College, Easton, PA 18042 USA (e-mail: BalanA@lafayette.edu; TraldiL@lafayette.edu).

Digital Object Identifier 10.1109/TR.2003.816403

[1]The singular and plural of an acronym are always spelled the same.

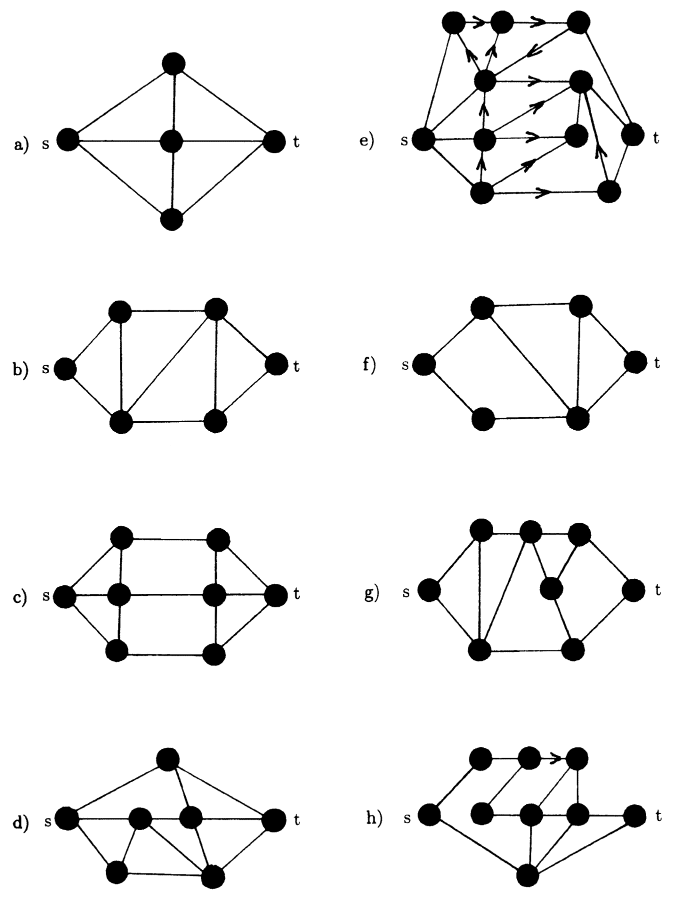

Fig. 1. Some networks for which smaller-minpaths-first preprocessing is not optimal.

This paper emphasizes the last distinction by discussing SVI-SDP algorithms first and MVI-SDP algorithms separately, later in the paper.

Example 2.1: The system has minpaths abc, def, $abdf$, and $acde$. An SVI-SDP analysis using this order of the minpaths creates 6 disjoint terms: abc, $\bar{a}def$, $a\bar{b}def$, $ab\bar{c}def$, $ab\bar{c}d\bar{e}f$, $a\bar{b}cde\bar{f}$.

If the minpaths are listed in the order abc, $abdf$, def, $acde$ then only 5 disjoint terms are needed: abc, $ab\bar{c}df$, $\bar{a}def$, $a\bar{b}def$, $a\bar{b}cde\bar{f}$. ◀

This example illustrates that SVI-SDP algorithms do not function efficiently when disjoint (or almost disjoint) minpaths appear early in the minpath list. Other examples of this inefficiency are easy to find, *e.g.*, it is proven [12] that every source-to-terminal network in Fig. 1 has the property that 'all of its optimal SVI-SDP analyses violate the preprocessing suggestion of [1]'. (Here networks with perfect nodes are considered; thus only the links appear in the minpaths.) This nonoptimality of the advice of [1] is fairly common among source-to-terminal networks. The first 12 examples in [11], and the 3 networks obtained from these examples by series and parallel reductions, were studied. Each of these 15 networks has an optimal SVI-SDP analysis which disregards the advice of [1] at least once; only the 7 networks not pictured in Fig. 1 may also have optimal SVI-SDP analyses which follow this advice.

There are 9 terms required to negate $abc + def$: in each negating term, one of a, b, c is negated and one of d, e, f is negated. Only 8 of these terms are required to negate $abc + def + abdf$, because $\bar{c}\bar{e}$ does not negate $abdf$. It might seem that negating $abc + def + abdf + acde$ requires only 7 terms, because $\bar{c}\bar{e}$ and $\bar{b}\bar{f}$ are removed from the original 9. However, 8 disjoint terms are actually required for this negation: $\bar{a}\bar{d}+\bar{a}d\bar{e}+\bar{a}de\bar{f}+a\bar{b}\bar{d}+a\bar{b}d\bar{e}+ab\bar{c}\bar{d}+ab\bar{c}d\bar{f}+a\bar{b}\bar{c}de\bar{f}$, for instance. See [5], [10], and Section IV for a discussion of this kind of process.

Example 2.1: The system of example 2.1 is duplicated n times, using n separate alphabets $a_j, b_j, c_j, d_j, e_j, f_j$. Let an SDP analysis be performed after the minpaths are listed in the order $a_1b_1c_1$, $a_1b_1d_1f_1$, $d_1e_1f_1$, $a_1c_1d_1e_1$, $a_2b_2c_2$, $a_2b_2d_2f_2, \ldots, d_ne_nf_n$, $a_nc_nd_ne_n$. If $2 \leq j \leq n$, then the 4 minpaths involving a_j, b_j, c_j, d_j, e_j, f_j give rise to $8^{j-1} \cdot 5$ disjoint terms, obtained by:

- choosing from the 8 negating terms of $a_ib_ic_i + d_ie_if_i + a_ib_id_if_i + a_ic_id_ie_i$ for each $i < j$,
- also choosing from $a_jb_jc_j$, $a_jb_j\bar{c}_jd_jf_j$, $\bar{a}_jd_je_jf_j$, $a_j\bar{b}_jd_je_jf_j$, $a_j\bar{b}_jc_jd_je_j\bar{f}_j$.

Five disjoint terms are required for $a_1b_1c_1$, $a_1b_1e_1f_1$, $d_1e_1f_1$, $b_1c_1d_1e_1$. Consequently the SDP analysis results in $5 \cdot \sum_{j=1}^{n} 8^{j-1}$ disjoint terms, fewer than $6 \cdot 8^{n-1}$ disjoint terms.

Consider an SDP analysis performed after the minpaths are preprocessed with the 3-element minpaths listed first. 3^{2n-1} disjoint terms are required for the last 3-element minpath; each disjoint term involves choosing 1 variable from each of the preceding minpaths to be negated. Similarly, the next-to-last 3-element minpath requires 3^{2n-2} disjoint terms, and the one before that requires 3^{2n-3} disjoint terms. Let the first 4-element minpath be $a_1b_1d_1e_1$; it requires 9^{n-1} disjoint terms, obtained by choosing one of the 9 negating terms of $a_jb_jc_j + d_je_jf_j$ for each $j > 1$. If the second 4-element minpath is $a_1c_1d_1e_1$, it too requires 9^{n-1} disjoint terms. If, instead, the second 4-element minpath is, say, $a_2b_2d_2f_2$, then it requires only $8 \cdot 9^{n-2}$ disjoint terms because $a_1b_1c_1 + a_1b_1d_1f_1 + d_1e_1f_1$ has only 8 negating terms. Thus there must be at least $3^{2n-1} + 3^{2n-2} + 3^{2n-3} + 9^{n-1} + 8 \cdot 9^{n-2} = 9^{n-2} \cdot (27+9+3+9+8) > 9^{n-2} \cdot 54 = 9^{n-1} \cdot 6$ disjoint terms arising from these 5 minpaths.

Each of the $2n - 2$ remaining 4-element minpaths gives rise to $8^a \cdot 9^b$ disjoint terms for some a and b with $a + b = n - 1$: if the minpath uses alphabet $\#j$, then its disjoint terms involve choosing 1 out of 8 or 9 negating terms for each $i \neq j$. Hence these minpaths give rise to at least $(2n-2) \cdot 8^{n-1}$ disjoint terms. In total, there are more than $6 \cdot 9^{n-1} + (2n - 2) \cdot 8^{n-1}$ disjoint terms.

Thus conventional preprocessing, with smaller minpaths listed first, leads to an SDP analysis which involves at least

$$\frac{6 \cdot 9^{n-1} + (2n-2) \cdot 8^{n-1}}{6 \cdot 8^{n-1}} = \left(\frac{9}{8}\right)^{n-1} + \frac{n-1}{3}$$

times as many disjoint terms as are actually necessary, no matter which minpath-based SVI-SDP algorithm is used. For instance, when $n = 3$, an implementation of the algorithm of [1] yields 850 disjoint terms after smaller-minpaths-first preprocessing, and only 365 disjoint terms when the other order of the minpaths is used. ◀

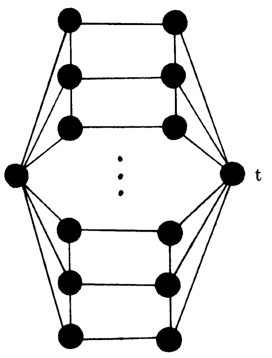

Fig. 2. Ladder network.

Essentially, the number of disjoint terms in example 2.2 grows with 9^n when the preprocessing suggestion of [1] is followed, and only with 8^n when the minpaths are listed in the mixed-size order. The same kind of disparity applies to other similar examples, *e.g.*, the system with minpaths $a_1b_1c_1$, $b_1c_1d_1e_1$, $d_1e_1f_1$, $e_1f_1a_2b_2$, $a_2b_2c_2$, $b_2c_2d_2e_2, \ldots, b_nc_nd_ne_n$, $d_ne_nf_n$ or the 'ladder network' with $2n$ 'rungs' in Fig. 2.

TABLE I
A 54-TERM ANALYSIS OF THE EXAMPLE IN [1]

jkl					
$bcjl$	$\bar k$				
$acfjl$	$\bar b\bar k$				
$dfhk$	$\bar j$	$j\bar l$			
$acdh$	$\bar f\ \bar j$	$\bar f j\bar l$	$\bar f jl\bar k\bar b$	$f\bar k\bar j$	$f\bar k j\bar l$
$abdhk$	$\bar c\bar f\ \bar j$	$\bar c\bar f j\bar l$			
$bcdfh$	$\bar a\bar k\bar j$	$\bar a\bar k j\bar l$			
$ghijk$	$\bar l\bar d$	$\bar l d\bar f\bar a$	$\bar l d\bar f a\bar b\bar c$		
$acegh$	$\bar d\ \bar j$	$\bar d j\bar k\bar l$	$\bar d j\bar k l\bar b\bar f$	$\bar d jk\bar\imath\bar l$	
$aceil$	$\bar j\bar h$	$\bar j h\bar d\bar g$	$j\bar b\bar f\bar k\bar h$	$j\bar b\bar f\bar k h\bar d\bar g$	
$efghk$	$\bar d\ \bar j\bar a$	$\bar d\ \bar j a\bar c$	$\bar d j\bar\imath\bar l\bar a$	$\bar d j\bar\imath\bar l a\bar c$	
$efikl$	$\bar j\bar a\bar h$	$\bar j\bar a h\bar d\bar g$	$\bar j a\bar c\bar h$	$\bar j a\bar c h\bar d\bar g$	
$abeghk$	$\bar c\bar d\bar f\bar j$	$\bar c\bar d\bar f j\bar\imath\bar l$			
$bcefgh$	$\bar a\bar d\ \bar k\bar j$	$\bar a\bar d\ \bar k j\bar l$			
$dehijk$	$\bar f\bar g\bar l\bar a$	$\bar f\bar g\bar l a\bar b\bar c$			
$abeikl$	$\bar c\bar f\ \bar j\bar h$	$\bar c\bar f\ \bar j h\bar d\bar g$			
$acdgil$	$\bar e\bar h\bar j$	$\bar e\bar h j\bar b\bar f\bar k$			
$bcefil$	$\bar a\bar j\bar k\bar h$	$\bar a\bar j\bar k h\bar d\bar g$			
$bcghij$	$\bar k\bar l\bar a\bar f$	$\bar k\bar l\bar a f\bar d\bar e$	$\bar k\bar l a\bar d\bar e$		
$dfgikl$	$\bar e\bar h\bar j\bar a$	$\bar e\bar h\bar j a\bar c$			
$acfghij$	$\bar b\bar d\bar e\bar k\bar l$				
$bcdehij$	$\bar a\bar f\bar g\bar k\bar l$				
$abdgikl$	$\bar c\bar e\bar f\bar h\bar j$				
$bcdfgil$	$\bar a\bar e\bar h\bar j\bar k$				

III. ABRAHAM'S EXAMPLE

The examples in Section II strongly suggest that when preprocessing a list of minpaths for an SVI-SDP algorithm, it is desirable to avoid the appearance of minpaths which are largely disjoint from all their predecessors in the list. This suggestion can also be considered in connection with the most-studied example in the references, a system with 12 elements and 24 minpaths mentioned in [1], and pictured in Fig. 1(g). (Table I gives a complete list of the minpaths.) The algorithm of [1] provides an SVI-SDP analysis of this example with 71 disjoint terms. [6] modifies the algorithm of [1] in 2 ways:

1) listing the minpaths of a given size lexicographically,
2) improving the portion of the algorithm which generates negating terms.

The result is a more efficient SVI-SDP analysis with 61 disjoint terms [7]. Both [1] and [6] list the disjoint minpaths jkl and $acdh$ at the beginning of the list. [13] introduces a preprocessing scheme which is based (as observed in [11]) on Hamming distances, and produces an SVI-SDP analysis of the example of [1] which involves only 59 disjoint terms [7]. As noted in [13], the improvement arises from the re-ordering at the beginning of the minpath list, which [13] starts as jkl, $bcjl$, $acdh$, avoiding the consecutive appearance of disjoint minpaths.

Reference [3] gives a further improvement, reducing the number of disjoint terms to 55 with a minpath list that starts jkl, $bcjl$, $dfhk$, $acdh$, $abdhj$, $acfjl$.

That preprocessing-strategy lists the minpaths of a given size in descending order of an intersection score: the average number of elements a minpath shares with smaller minpaths. (This is the same as listing first those that have the smallest average relative complements within the smaller minpaths, which is a sensible strategy because these relative complements contribute to the disjoint terms that arise from the minpaths being considered.) This score does not suggest a complete order in which to list the minpaths of a given size, because different minpaths can be tied with the same score. For the example of [1], other choices of lists consistent with the preprocessing strategy of [3] can result in SVI-SDP analyses involving 54–60 disjoint terms.

Two observations about the list in [3] are:

1) [3, table 2] contains only 53 of the 55 disjoint terms. The disjoint terms arising from the minpath $aceil$ should include $ac\bar d e\bar g hi\bar j l$ (or 1-101-0110-1 in the notation of [3]) and the disjoint terms arising from $efikl$ should include $\bar a ef\bar h i\bar j kl$ (0- - -11-01 011). There is also a 0 missing at the end of the fourth disjoint term. (See Section IV.)
2) The number of disjoint terms can be reduced if $acfjl$ is placed either before or after $dfhk$ in the list, because $acfjl$ requires 2 disjoint terms in its current position but requires only 1 if it is moved up in the list, and moving it does not increase the number of disjoint terms required for any other minpath. Moving $acfjl$ is suggested by the principle in Section II (disjointness should be avoided early in the list of minpaths) because $acfjl$ intersects all of the 3- and 4-element minpaths. It is striking that this move improves the best published SVI-SDP analysis of the example of [1] by disregarding the preprocessing advice of [1], placing a 5-element minpath ahead of some of the 4-element minpaths.

Table I gives the complete SVI-SDP analysis of the example in [1] that arises when the order of [3] is modified by listing $acfjl$ third instead of sixth. The minpaths are listed in column #1, with the corresponding negating terms listed to their right. The variables in the negating terms are not given alphabetically, but in a way that suggests the pivoting method used to produce the disjoint terms.

IV. Checking SDP Calculations

[3], [6], [8], [13] contain mistaken SDP analyses of the example in [1]. For the convenience of those who might want occasionally to perform such calculations, this discussion of preprocessing is interrupted to discuss ways to check the accuracy (though not the minimality) of a set of disjoint terms supposed to describe a product $P_j \cdot \bar{P}_{j-1}, \ldots, \bar{P}_1$. The focus is on the analysis of [3]; see [7] for a discussion of [6] and [13].

1) Check to see if the disjoint terms are actually disjoint. This is enough to catch one of the errors in [3, table 2], the missing 0 at the end of the fourth disjoint term.
2) Check to see if P_j actually appears in one of the disjoint terms that are supposed to represent $P_j \cdot \bar{P}_{j-1}, \ldots, \bar{P}_1$. This is enough to catch another error, because none of the disjoint terms contain the minpath $efikl$.
3) Consider the minpath $aceil$ in the example of [1], which appears tenth in the list suggested in [3]: $jkl = P_1$, $bcjl = P_2$, $dfhk = P_3$, $acdh = P_4$, $abdhk = P_5$, $acfjl = P_6$, $bcdfh = P_7$, $ghijk = P_8$, $acegh = P_9$, $aceil = P_{10}$.

Following [5] and [10], calculate $P_{10} \cdot \bar{P}_9, \ldots, \bar{P}_2 \cdot \bar{P}_1$ by Boolean multiplication of the relative complements $P_1 - P_{10}, \ldots, P_9 - P_{10}$, with absorption.

$$\begin{aligned}
&(\bar{j}+\bar{k})\cdot(\bar{b}+\bar{j})\cdot(\bar{d}+\bar{f}+\bar{h}+\bar{k})\cdot(\bar{d}+\bar{h})\\
&\quad\cdot(\bar{b}+\bar{d}+\bar{h}+\bar{k})\\
&\quad\cdot(\bar{f}+\bar{j})\cdot(\bar{b}+\bar{d}+\bar{f}+\bar{h})\cdot(\bar{g}+\bar{h}+\bar{j}+\bar{k})\\
&\quad\cdot(\bar{g}+\bar{h})\\
&\;=(\bar{j}+\bar{k})\cdot(\bar{b}+\bar{j})\cdot(\bar{d}+\bar{h})\cdot(\bar{f}+\bar{j})\cdot(\bar{g}+\bar{h})\\
&\;=(\bar{j}+\bar{b}\bar{k})\cdot(\bar{f}+\bar{j})\cdot(\bar{d}+\bar{h})\cdot(\bar{g}+\bar{h})\\
&\;=(\bar{j}+\bar{b}\bar{f}\bar{k})\cdot(\bar{h}+\bar{d}\bar{g})\\
&\;=\bar{h}\bar{j}+\bar{d}\bar{g}\bar{j}+\bar{b}\bar{f}\bar{h}\bar{k}+\bar{b}\bar{d}\bar{f}\bar{g}\bar{k}.
\end{aligned} \tag{4.1}$$

These equations give rise to 4 disjoint terms, $\bar{h}\bar{j}$, $\bar{d}\bar{g}h\bar{j}$, $\bar{b}\bar{f}\ \bar{h}j\bar{k}$, $\bar{b}\bar{d}\bar{f}\bar{g}hj\bar{k}$.

To check the accuracy of the calculation, also perform an SDP analysis of the binary coherent system given by the relative complements: jk, bj, $dfhk$, dh, $bdhk$, fj, $bdfh$, $ghjk$, gh; thus the minpaths are jk, bj, fj, dh, gh. An SDP analysis is jk, $bj\bar{k}$, $\bar{b}fj\bar{k}$, $dh\bar{j}$, $\bar{b}d\bar{f}hj\bar{k}$, $\bar{d}gh\bar{j}$, $\bar{b}\bar{d}\bar{f}ghj\bar{k}$. It is not difficult to check that these terms are actually disjoint from each other and from the 4 negating terms in the preceding paragraph.

If these calculations are correct, then

$$\begin{aligned}
&\bar{h}\bar{j}+\bar{d}\bar{g}h\bar{j}+\bar{b}\bar{f}\bar{h}j\bar{k}+\bar{b}\bar{d}\bar{f}\bar{g}hj\bar{k}+jk+bj\bar{k}\\
&\qquad+\bar{b}fj\bar{k}+dh\bar{j}+\bar{b}d\bar{f}hj\bar{k}+\bar{d}gh\bar{j}+\bar{b}\bar{d}\bar{f}ghj\bar{k},
\end{aligned} \tag{4.2}$$

provides a disjoint analysis of the space of all possible assignments of truth values (T, F) to the 7 variables b, d, f, g, h, j, k. There are 2^7 such assignments. Each disjoint term covers 2^{7-p} assignments, where p is the number of variables that appear in that disjoint term; *e.g.*, $bj\bar{k}$ covers 2^4 assignments. Hence the analysis of $P_{10} \cdot \bar{P}_9, \ldots, \bar{P}_2 \cdot \bar{P}_1$ is checked by verifying:

$$\begin{aligned}
&2^5+2^3+2^2+2^0+2^5+2^4\\
&\qquad+2^3+2^4+2^1+2^3+2^0=2^7.
\end{aligned} \tag{4.3}$$

The MVI-SDP analyses may be checked using similar steps; *e.g.*, a check of [8, table 2] indicates that disjoint terms #12 and #13 are not actually disjoint. (Correcting the error does not require an increase in the number of disjoint terms.) When using step 3 to check an MVI-SDP analysis, remember that a disjoint MVI term which involves variable groupings of sizes $q_1, \ldots, q_k$ covers $2^r \cdot (2^{q_1}-1) \cdot \cdots \cdot (2^{q_k}-1)$ truth value assignments, where r is the number of variables that are not mentioned in that disjoint term.

V. A New Preprocessing Strategy

Reference [3] predicates its preprocessing on the advice of [1], as do all the other papers mentioned in the references. The examples discussed in this paper suggest, instead, a preprocessing strategy that allows violations of the advice of [1] on occasion. One way to do this is to use dynamic-preprocessing: begin with jkl, and at each iteration choose the next minpath in the list so that it will have the best possible score according to some formula which seems appropriate. A reasonable score is the average size of the relative complements with the minpaths that have already been chosen: after $P_1, \ldots, P_{j-1}$ have been chosen, one might anticipate that the smaller the average size of the relative complements $P_1 - P_j, \ldots, P_{j-1} - P_j$, then the smaller the number of disjoint terms required to analyze the event $P_j \cdot \bar{P}_{j-1} \cdot \cdots \cdot \bar{P}_1$ will be. This strategy seems successful initially, because it does a good job of choosing one of the available minpaths which requires a small number of disjoint terms. However, it does not attempt to choose one of the available minpaths which will contribute to creating a small number of disjoint terms for the remaining minpaths, so it tends to produce SDP analyses with large numbers of disjoint terms for minpaths near the end of the list.

To compensate for this failing, modify this score to incorporate the advice of [1], though not as a binding requirement. One way to do this is to weigh the relative complements which come from smaller already-listed minpaths more heavily in the average; another way is to give smaller candidate minpaths an advantage. Both of these modifications are used if, with $P_1, \ldots, P_{j-1}$ already listed, P_j is chosen so that

$$\frac{2^{|P_j|}}{|P_j|} \cdot \sum_{k=1}^{j-1} \left(\frac{|P_k - P_j|}{|P_k|} \right) \tag{5.1}$$

is as small as possible. This scoring system has performed better than several other systems on a variety of examples.

The sum in (5.1) gives a weighted average of the relative complements, favoring those which come from smaller already-listed minpaths; it is unnecessary to divide by $j-1$ in this 'average' because $j-1$ is the same for all candidates. The factor $2^{|P_j|}/|P_j|$ gives an advantage to smaller candidate minpaths. It might seem to be a complicated assessment of the advantage smaller minpaths should have, but it is not impossible to rationalize: the $2^{|P_j|}$ reflects the subsets of the candidate minpath which can serve as relative complements for later terms, and dividing by $|P_j|$ indicates an averaging of the numbers of appearances of the elements of P_j in earlier minpaths.

There are 3 important details of the implementation of this system:

1) Adjust the factor $2^{|P_j|}/|P_j|$ to increase the advantage given to the smallest minpaths:
   - when $|P_j| = 1$, replace the factor $2^{|P_j|}/|P_j|$ with 0 (to guarantee that 1-element minpaths are always listed first),
   - when $|P_j| = 2$, replace the factor $2^{|P_j|}/|P_j|$ with 1.5 (to give 2-element minpaths an increased advantage).
2) An element of E which appears in all the minpaths not already listed cannot be negated in the remaining portion of the SDP analysis, and therefore will not affect the number of disjoint terms required. Such an element should be ignored in calculating $|P_j|$.
3) The scoring system does not seem to apply to the choice of the first minpath in the list, because (5.1) refers to the already-listed minpaths $P_1, \ldots, P_{j-1}$. To choose the first minpath, score each minpath according to the same formula, but using all the other minpaths in place of $P_1, \ldots, P_{j-1}$. (In the example of [1], jkl is the minpath with the lowest such score.)

This strategy is similar to the preprocessing strategy of [3] in several ways. Each strategy can have ties, so neither completely determines a list of the minpaths. However this new strategy usually has many fewer ties than the strategy of [3] (which in turn has many fewer ties than the strategy of [1]). In choosing which minpath should be listed next, each strategy has 2 goals:

1) keep the number of disjoint terms needed for the next minpath itself relatively low;
2) keep the next minpath's contribution to 'the numbers of disjoint terms that later-listed minpaths will require' relatively low.

Each strategy considers a simple average to address goal #1, and the advice of [1] to address goal #2. For each strategy, both of these considerations are compromises between precision and computation-economy:

a) on the one hand, a detailed analysis of the relative complements would generally give much more information (*e.g.*, consider the difference between having 2 disjoint 3-element relative complements and having the same 3-element relative complement occur twice),
b) on the other hand, a simpler preprocessing strategy (like the 'follow the advice of [1] and list minpaths of each size randomly' strategy mentioned in [11]) would run much more quickly.

A difference between the 2 strategies is that this one finds the desirable mixed-size lists for the examples of Section II, unlike any preprocessing strategy that does not allow any exceptions to the advice of [1].

In the example of [1] this new strategy suggests only one list—there are no ties. This list gives rise to the SDP analysis in Table II, with only 53 disjoint terms, fewer than any other published SVI-SDP analysis of this example. The exhaustive (and exhausting!) analysis required to determine whether or not Table II gives the best possible SVI-SDP analysis of the example of [1] has not been done, but it has been proven [12] that every SVI-SDP analysis of the example of [1] which follows the preprocessing advice of [1] involves at least 54 disjoint terms.

TABLE II
A 53-TERM ANALYSIS OF THE EXAMPLE IN [1]

jkl						
$bcjl$	$\bar{k}$					
$acfjl$	$\bar{b}\bar{k}$					
$acdh$	$\bar{j}$	$j\bar{l}$	$jl\bar{b}\bar{f}\bar{k}$			
$dfhk$	$\bar{c}\bar{j}$	$\bar{c}j\bar{l}$	$c\bar{a}\bar{j}$	$c\bar{a}j\bar{l}$		
$bcdfh$	$\bar{a}\bar{k}\bar{j}$	$\bar{a}\bar{k}j\bar{l}$				
$abdhk$	$\bar{c}\bar{f}\ \bar{j}$	$\bar{c}\bar{f}j\bar{l}$				
$ghijk$	$\bar{l}\bar{d}$	$\bar{l}d\bar{f}\bar{a}$	$\bar{l}d\bar{f}a\bar{b}\bar{c}$			
$efghk$	$\bar{d}\ \bar{j}$	$\bar{d}j\bar{\imath}\bar{l}$				
$acegh$	$\bar{d}\ \bar{j}\bar{f}$	$\bar{d}\ \bar{j}f\bar{k}$	$\bar{d}j\overline{lk}$	$\bar{d}j\bar{l}k\bar{f}\bar{\imath}$	$\bar{d}jlk\bar{b}\bar{f}$	
$efikl$	$\bar{j}\bar{h}$	$\bar{j}h\bar{d}\bar{g}$				
$aceil$	$\bar{h}\bar{k}\bar{j}$	$\bar{h}\bar{k}j\bar{b}\bar{f}$	$\bar{h}k\overline{fj}$	$h\bar{g}\bar{d}\ \bar{j}\bar{f}$	$h\bar{g}\bar{d}\ \bar{j}f\bar{k}$	$h\bar{g}\bar{d}j\bar{b}\bar{f}\bar{k}$
$dehijk$	$\bar{f}\bar{g}\bar{l}\bar{a}$	$\bar{f}\bar{g}\bar{l}a\bar{b}\bar{c}$				
$abeghk$	$\bar{c}\bar{d}\bar{f}\ \bar{j}$	$\bar{c}\bar{d}\bar{f}j\bar{\imath}\bar{l}$				
$bcefgh$	$\bar{a}\bar{d}\ \bar{k}\bar{j}$	$\bar{a}\bar{d}\ \bar{k}j\bar{l}$				
$bcghij$	$\bar{k}\bar{l}\bar{a}\bar{f}$	$\bar{k}\bar{l}\bar{a}f\bar{d}\bar{e}$	$\bar{k}\bar{l}a\bar{d}\bar{e}$			
$abeikl$	$\bar{c}\bar{f}\ \bar{j}\bar{h}$	$\bar{c}\bar{f}\ \bar{j}h\bar{d}\bar{g}$				
$bcefil$	$\bar{a}\bar{j}\bar{k}\bar{h}$	$\bar{a}\bar{j}\bar{k}h\bar{d}\bar{g}$				
$dfgikl$	$\bar{e}\bar{h}\bar{j}$					
$acdgil$	$\bar{e}\bar{h}\bar{j}\bar{f}$	$\bar{e}\bar{h}\bar{j}f\bar{k}$	$\bar{e}\bar{h}j\bar{k}\bar{b}\bar{f}$			
$acfghij$	$\bar{b}\bar{d}\bar{e}\bar{k}\bar{l}$					
$bcdehij$	$\bar{a}\bar{f}\bar{g}\bar{k}\bar{l}$					
$bcdfgil$	$\bar{a}\bar{e}\bar{h}\bar{j}\bar{k}$					
$abdgikl$	$\bar{c}\bar{e}\bar{f}\bar{h}\bar{j}$					

When applied to the 15 networks drawn from [10, figures 1–12] which have been studied, this preprocessing system appreciably outperforms the advice of [1]. For every one of the 15 networks, this system proposes at least 1 minpath list which gives rise to an SVI-SDP analysis with as few disjoint terms as any SVI-SDP analysis which follows the advice of [1]. For 7 of the 15 networks, this system proposes minpath lists which give rise to SVI-SDP analyses with fewer disjoint terms than all SVI-SDP analyses which follow the advice of [1]; these 7 are in Fig. 1(a)–(g). Because of the small number of ties this system proposes only a few possible lists for each network, many fewer than the advice of [1] allows. Do not get the impression that this preprocessing system is perfect; it is not; *e.g.*, it incorrectly recommends disregarding the advice of [1] for the system with minpaths ab, bd, ce, acd, ade. Also, it fails to find optimal minpath lists for at least 3 of the 15 networks, those shown in Fig. 1(c), (e) and (h).

VI. Some Examples for MVI-SDP

The preprocessing strategy suggested in [1] is usually used in both SVI-SDP and MVI-SDP algorithms. This section presents examples for which the optimal MVI-SDP preprocessing violates the advice of [1]. In each of these examples, the optimal preprocessing for SVI-SDP follows the suggestion of [1]; thus these examples also illustrate the fact that optimal preprocessing for SVI-SDP and MVI-SDP algorithms is different. This difference explains some of the results of [11], which observes that in many examples the preprocessing strategies devised in [6], [7], [13] are no more effective for an MVI-SDP algorithm than the advice of [1] alone.

Example 6.1: The system has minpaths $defg$, abc, chi. An MVI-SDP analysis using this order of the minpaths is

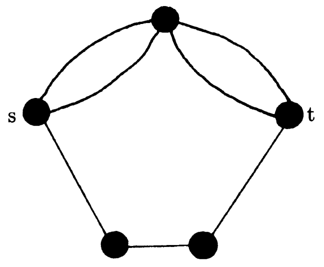

Fig. 3. A series-parallel network.

$defg$, $\overline{defg}abc$, $\overline{ab}c\overline{defg}hi$. However if the minpaths are listed abc, chi, $defg$, then the resulting MVI-SDP analysis involves 4 terms: abc, $\overline{ab}chi$, $\bar{c}defg$, $\overline{ab}cdefg\overline{hi}$. The list abc, chi, $defg$ is optimal for SVI-SDP, because it avoids the appearance of disjoint minpaths at the beginning of the list. ◀

For the purposes of MVI-SDP, the minpaths of Example 6.1 are 'really' just d, ac, ch, because every other variable is 'really the same' as one of a, c, d, h. The optimal minpath list reflects the accuracy of the advice of [1] for these 'real' minpaths.

Example 6.2: The system of Example 6.1 is duplicated n times using n separate alphabets. Observe that abc, $defg$, chi can be negated with 2 disjoint MVI terms: $\bar{c}\overline{defg}$ and $\overline{ab}c\overline{defg}\,\overline{hi}$. Also, negating abc, chi requires 2 disjoint MVI terms: $\bar{c}$ and $\overline{ab}c\overline{hi}$. If the minpaths are preprocessed conventionally, then the last n minpaths in the list are $d_1e_1f_1g_1$, $d_2e_2f_2g_2$, \ldots , $d_ne_nf_ng_n$. Each $d_je_jf_jg_j$ is represented by 2^n disjoint terms, choosing 1 negating term from each pair $(\bar{c}_k\overline{d_ke_kf_kg_k},\ \overline{a_kb_k}c_k\overline{d_ke_kf_kg_k}\ \overline{h_ki_k})$ with $k < j$, and 1 negating term from each pair $(\bar{c}_k,\ \overline{a_kb_k}c_k\overline{h_ki_k})$ with $k \geq j$. There are more than $n \cdot 2^n$ disjoint terms, in all.

On the other hand, let the minpaths be listed in the order $a_1b_1c_1,\ a_2b_2c_2, \ldots, a_nb_nc_n,\ d_1e_1f_1g_1, \ldots, d_ne_nf_ng_n,$ $c_1h_1i_1, \ldots, c_nh_ni_n$. Then each of the first $2n$ minpaths requires only a single MVI term, and each $c_jh_ji_j$ requires 2^{j-1} disjoint terms, because there are 2 negating terms for each $i < j$. Consequently this list gives rise to an MVI-SDP analysis with $2n + \sum_{j=1}^{n} 2^{j-1}$ disjoint terms, fewer than $2 \cdot 2^n$.

Conventional preprocessing leads to an MVI-SDP analysis which involves more than $n/2$ times as many disjoint terms as are actually necessary, no matter what MVI-SDP algorithm is used. ◀

Example 6.3: For the network in Fig. 3, the advice of [1] gives the worst possible minpath list for MVI-SDP, although it is optimal for SVI-SDP. The optimal minpath list for MVI-SDP is obtained by thinking of the 3 series elements as 'really just 1 element', and listing the 3-element minpath first. ◀

Examples 6.1–6.3 illustrate the fact that all available series reductions should be performed before an MVI-SDP analysis. However not all examples of nonoptimality of the advice of [1] for MVI-SDP are associated with unperformed series reductions.

Example 6.4: The system has minpaths $abeh$, $cdfg$, aci, $abcdef$, $abcdgh$, $bdefghi$. An MVI-SDP analysis using this order of the minpaths is $abeh$, $\overline{abeh}cdfg$, $a\overline{beh}c\overline{dfg}i$, $abcdef\bar{g}\bar{h}\bar{i}$, $abcd\bar{e}\bar{f}gh\bar{i}$, $\bar{a}b\bar{c}defghi$. If the minpaths are listed with aci first, however, it is not possible to have only 1 disjoint term for each minpath: which one of $abeh$ or $cdfg$ comes after the other, requires at least 2 disjoint terms. In this example, too, a conventionally preprocessed list is optimal for SVI-SDP. ◀

It seems difficult to formulate a preprocessing system like that of Section V that works well for MVI-SDP algorithms, because it is difficult to know exactly what to count in a formula: the individual elements of a system are separate in some parts of an MVI-SDP analysis, and together in others. It is clear, however, that any possible series and parallel reductions should be performed before preprocessing.

VII. Limitations on the Effectiveness of Preprocessing

Not all examples of nonoptimal performance of SDP algorithms result from preprocessing choices; there are systems for which there is no preprocessing that results in optimal SDP analyses.

Example 7.1: Let n be a positive integer, and consider a system with 3 disjoint minpaths: $A = a_1 \ldots a_n$, $B = b_1 \ldots b_n$, $C = c_1 \ldots c_n$. Preprocessing does not affect the number of disjoint terms produced by an SVI-SDP algorithm, $n^2 + n + 1$. However the system has an SVI-SDP analysis with only $3n + 1$ disjoint terms: $\bar{a}_1B,\ a_1\bar{a}_2B, \ldots, a_1 \ldots a_{n-1}\bar{a}_nB,$ $\bar{b}_1C, \ldots, b_1 \ldots b_{n-1}\bar{b}_nC,\quad \bar{c}_1A, \ldots, c_1 \ldots c_{n-1}\bar{c}_nA,\quad ABC$. Thus all SVI-SDP algorithms analyze this system using $O(n)$ times as many disjoint terms as are actually necessary. ◀

Similar examples can be given for which all SVI-SDP algorithms give analyses with up to $O(n^{11})$ times the minimum number of disjoint terms. There are also systems for which all MVI-SDP algorithms overestimate the required number of disjoint terms by factors up to $O(n^9)$. The exponents 11 and 9 are not absolute ceilings but reflect the authors' limited ability to analyze these examples; see [12] for details.

References

[1] J. A. Abraham, "An improved method for network reliability," *IEEE Trans. Rel.*, vol. R-28, pp. 58–61, 1979.

[2] F. Beichelt and L. Spross, "An improved abraham method for generating disjoint sums," *IEEE Trans. Rel.*, vol. R-36, pp. 70–74.

[3] ——, "Comment on: An improved abraham method for generating disjoint sums," *IEEE Trans. Rel.*, vol. 38, pp. 422–424, 1989.

[4] K. D. Heidtmann, "Smaller sums of disjoint products by subproduct inversion," *IEEE Trans. Rel.*, vol. 38, pp. 303–311, 1989.

[5] M. O. Locks, "Inverting and minimalizing path sets and cut sets," *IEEE Trans. Rel.*, vol. R-27, pp. 107–109, 1978.

[6] ——, "A minimizing algorithm for sum of disjoint products," *IEEE Trans. Rel.*, vol. R-36, pp. 445–453, 1987.

[7] M. O. Locks and J. M. Wilson, "Note on disjoint product algorithms," *IEEE Trans. Rel.*, vol. 41, pp. 81–84, 1992.

[8] ——, "Nearly minimal disjoint forms of the abraham reliability problem," *Reliab. Eng. Syst. Saf.*, vol. 46, pp. 283–286, 1994.

[9] S. Rai, M. Veeraraghavan, and K. S. Trivedi, "A survey of efficient reliability computation using disjoint products approach," *Networks*, vol. 25, pp. 147–163, 1995.

[10] D. R. Shier and D. E. Whited, "Algorithms for generating minimal cutsets by inversion," *IEEE Trans. Rel.*, vol. R-34, pp. 314–319, 1985.

[11] S. Soh and S. Rai, "Experimental results on preprocessing of path/cut terms in sum of disjoint products technique," *IEEE Trans. Rel.*, vol. 42, pp. 24–33, 1993.

[12] L. Traldi, *Non-Minimal Sums of Disjoint Products*: Lafayette College, 2003.

[13] J. M. Wilson, "An improved minimizing algorithm for sum of disjoint products," *IEEE Trans. Rel.*, vol. 39, pp. 42–45, 1990.

Alexandru O. Balan graduated from the Tudor Vianu High School for Computer Science in Bucharest, Romania in July 1999, after winning Special Awards in the Romanian National Competition of Computer Programming in 1996 and 1998. He is studying Computer Science, Mathematics, and Economics at Lafayette College in Easton, Pennsylvania, where he is a member of the student chapter of the Association for Computing Machinery (ACM) in addition to Phi Beta Kappa and the honor societies for Computer Science, Economics, and Mathematics. He plans to attend graduate school in Computer Science at Brown University after finishing his Bachelor's degrees.

Lorenzo Traldi is a Professor of Mathematics at Lafayette College. He received the AB in 1976 in Mathematics from Queens College of the City University of New York, and the Ph.D. in 1980 in Mathematics from Yale University. His research interests include knots, graphs, and matroids in addition to network reliability. He is a member of the American Mathematical Society and the IEEE Reliability Society.