

失效树分析的新途径

廖 炯 生

(北京控制工程研究所)

摘 要

本文提出了布尔代数的不交型运算规则,用来从失效树直接写出不交型失效函数,计算失效概率,从而提供了失效树分析的新途径。它不用枚举最小割集或质蕴涵集,对相干和非相干系统可以统一处理,并把枚举程序和失效函数化不交型的程序简化为单一计算程序,节省了计算量。

新途径还对失效树进行分解和简约。为此提出“失效树的重复模块”概念,和对逻辑多余项目尽早简约的原则,进一步使计算量大为减少。

一、引 言

失效树分析法 (Fault Tree Analysis) 是分析复杂系统可靠性和安全性的有效手段。系统越复杂,系统失效的后果越严重,就越需要采用 FTA。

FTA 是以系统最不希望发生的事件(顶事件)作为分析的目标,找出系统内可能发生的元件失效、环境变化、人为失误以及程序处理等软件因素(各种底事件)与系统失效之间的逻辑联系,并用倒立树状图形表示出来。由此定性分析各底事件对系统失效发生影响的方式和途径(系统故障模式识别),以及定量计算这种影响的程度,由各底事件概率算出系统失效概率。

为了计算系统失效概率,要求出失效函数,并化成不交型积之和的表达式。现有的方法都是在推演成复杂的布尔函数(失效函数)之后才作不交性变换,因此费时费力。如能从失效树的基本环节(逻辑门)入手,直接化成不交型,那将省事得多,而且一劳永逸。这样做的关键在于建立布尔代数的不交型运算规则。

布尔函数最小化原则通常是使“交”、“并”运算总数最少,这意味着所使用的门电路总数最省,因而成为开关线路设计和简化的有力工具。但我们指出,在需要计算 0/1 事件发生概率的场合,采用一种新的在不交意义上最小化的原则是更方便的。例如一个双输入“OR”门,通常以 $A \cup B$ 为最小,但 $A \cup \bar{A}B$ 或 $B \cup \bar{B}A$ 对于概率计算更方便。二者的差别初看很小,但最终计算量可能差别很大。就像和事件

$$x_1 + x_2 + \cdots + x_n$$

的概率表达式可包含 $2^n - 1$ 项,而不交和事件

$$x_1 + \bar{x}_1 x_2 + \cdots + \bar{x}_1 \cdots \bar{x}_{n-1} \cdot x_n$$

的概率表达式仅有 n 项。

Cargal 最近指出^[1], FTA 的要害是太繁,采用一种高等代数是取得突破的最可能的途径。我们引入 Boolean 代数的不交型运算规则而得到成功。规则的实质就是 Boolean 函数在不交意义上的最小化。

Rosenthal 指出^[2], 求任意失效树的失效概率,和计算一般网络的可靠度一样,都是 NP 困难问题。现在有很强的理论上的和直觉的理由使大多数计算机科学家猜想: NP 问题在确定性计算机(现实计算机)上只有指数算法。因此对失效树进行分解(模块化)可能是最有希望的一条出路(在可能分解的限度内)。

为此,我们推广模块概念,提出“失效树的重复模块”概念,扩展了可能分解的范围,提高了分解效率,并提出对逻辑多余项目应尽早简约。

综合以上二方面,形成了失效树分析新途径。

二、失效树的分解和简约

在选定顶事件和建造失效树的基础上,应首先对失效树进行分解和简约。

失效树的模块是至少有二个事件的集合,它只有一个输出到达失效树的其余部分,而且没有来自其余部分的输入。这样的模块,可从失效树分割出来单独进行概率计算,而在失效树中用一个“准底事件”代替它^[3]。模块恒为相干系统,但相干系统(其失效函数是单调的)未必是模块。

我们把模块的概念推广,把失效树中的重复模块也分割出来,各用相同的“准底事件”代替。条件是重复模块对失效树其余部分和对其余模块必须没有重复事件。本文例 2 将对此作进一步说明。同时用割顶点(Cut Vertex)办法来找出可能的模块。

失效树的有效大小等于所有模块和重复模块各用“准底事件”代替后的剩余事件数目。这样就把规模压缩,计算量将按指数减少。

失效树的分解(或称析因),当然也是一种简约。另外,失效树内包含的逻辑多余项目也应当予以简约。一开始就进行这种简约,比问题展开以后再行简约要有效得多。人工建造失效树很繁,但能有力地推动分析人员去透彻地了解系统,所以现在还多用人工建造失效树。在这种情况下,人工检查一下是否包含逻辑多余项目,并予以简约是很方便的。简约规则是,根据失效树结构,检查底事件组合,取

$$x + xy \implies x, \quad xy + xy \implies xy, \quad xx \implies x.$$

具体参看本文例 2。

当然,失效树中逻辑多余项目的简约和模块搜索,也可以排成统一的程序,由计算机来完成。

三、Boolean 代数的不交型运算规则

我们对比地列出 Boolean 代数的一般运算规则和不交型运算规则,如表 1 所示。

记输出 Boolean 变量为 z , 输入变量为 A, B, C , 它们统计独立,但不必互斥。

根据概率加法定理和乘法定理易证,二种规则在概率计算方面完全等效。

表 1

	Boolean 代数运算规则	Boolean 代数的不交型运算规则
AND	$z = AB$	$z = AB$
OR	$z = A + B$	$z = A + \bar{A}B = B + \bar{B}A$
NAND	$z = \overline{AB} = \bar{A} + \bar{B}^{**}$	$z = \overline{AB} = \bar{A} + \bar{A}\bar{B} = \bar{B} + B\bar{A}^*$
NOR	$z = \overline{A + B} = \bar{A} \cdot \bar{B}^{**}$	$z = \overline{A + B} = \bar{A} \cdot \bar{B} = \bar{B} + \bar{B}A^*$ $z = \overline{AB + \bar{A}C} = \overline{AB} \cdot \overline{\bar{A}C}^*$
XOR	$z = A\bar{B} + \bar{A}B$ $z = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$	$z = A\bar{B} + \bar{A}B$ $z = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$
表决门	(以三中取二门为例) $z = AB + BC + CA$	$z = AB + \bar{A}BC + \bar{B}CA$ $= BC + \bar{B}CA + \bar{C}AB$ $= CA + \bar{C}AB + \bar{A}BC$

**——DeMorgan 定理, *——DeMorgan 定理的不交型。

同理可证,不交型表达式虽然不是唯一的,但在概率计算上是完全等效的。

运用 Boolean 代数的不交型运算规则,我们可以从失效树一下子写出不交型失效函数,再展开和吸收归并,得到不交型积之和的表达式。这样,就用不着再找最小割集或质蕴涵集了。因而可把枚举程序和化不交型的程序简化为单一计算程序,可以节省计算量。本文例 1 将进行对比说明。

FTA 的最小割集概念限于处理只由 AND, OR 门构成的相干失效树,对于带有“补”事件和 XOR 门的非相干失效树,还要引入质蕴涵集的概念。而我们这里定义了 AND, OR, NAND, NOR, XOR 及表决门的不交型运算规则,因而不相干非相干可以统一处理各种失效树。

不交型失效函数展开后应吸收归并,规则如下(虽然简约时检查了前三项,但写成失效函数展开后应再检查):

- 1) $x + xy \Rightarrow x$, 2) $xy + x\bar{y} \Rightarrow xy$, 3) $xx \Rightarrow x$, 4) $xy + x\bar{y} \Rightarrow x$, 5) $x\bar{x} \Rightarrow 0$.

符号“ $\Rightarrow$ ”表示归并的方向,约定不作反推。不把 $A + \bar{A}B$ 再化成 $A + B$ 。

这些运算都不包含决策判断,非常简单。失效函数经这样处理成不交型积之和的表达式,可以始终保持不交性,因而可用于最后计算系统失效概率。

之所以能够始终保持不交性,是因为我们从失效树写出失效函数时,从每一个逻辑门一直写到底事件或“准底事件”都是采用不交型运算规则。即使事件有重复(相交),也可以保证结果的不交性。然后的展开和吸收归并显然不引入相交的因素,所以最后结果为不交型。

不交性定理^[4]指出,如交集 T_1 中至少有一“字母”(事件) A 以“补”的形式出现在 T_2 中,则 T_1, T_2 代表不交群。据此易直观检查最后结果的不交性。不交型运算规则可用来简化网络可靠度计算。对此将另文讨论。

四、从不交型失效函数反求最小割集

如上所述,新途径不用枚举最小割集,但是最小割集在定性说明相干失效树底事件或“准底事件”对系统失效发生影响的方式和途径方面,还很有用。所以我们提出反求最小割集的方法。由于用不交型运算规则,从失效树的所有逻辑门直到所有底事件,写出不交型失效函数,所以它包含了所有的最小割集的信息在内。反求最小割集十分简便。

对于只由 AND, OR 门组成的相干失效树,输入中没有“补”,失效函数中所有“补”事件都是为了不交性而添入的,所以:

对于只由 AND, OR 门组成的相干失效树,输入中没有“补”,失效函数中所有“补”事件都是为了不交性而添入的,所以:

i) 从不交型失效函数所有各项取消所有“补”事件即得到各个割集。

ii) 再按 $x + x \Rightarrow x, x + xy \Rightarrow x$ 的规则吸收归并,即可得到所有的最小割集。

在失效树分解为模块的情况下,求得的是“模块最小割集”,它甚至比“事件最小割集”更

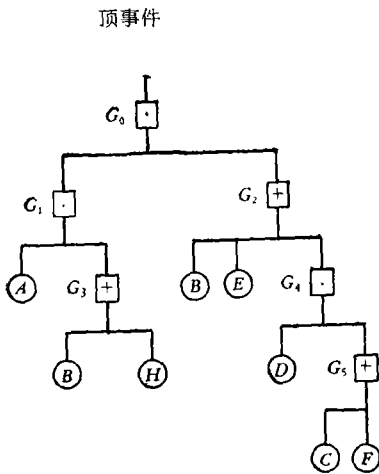


图 1

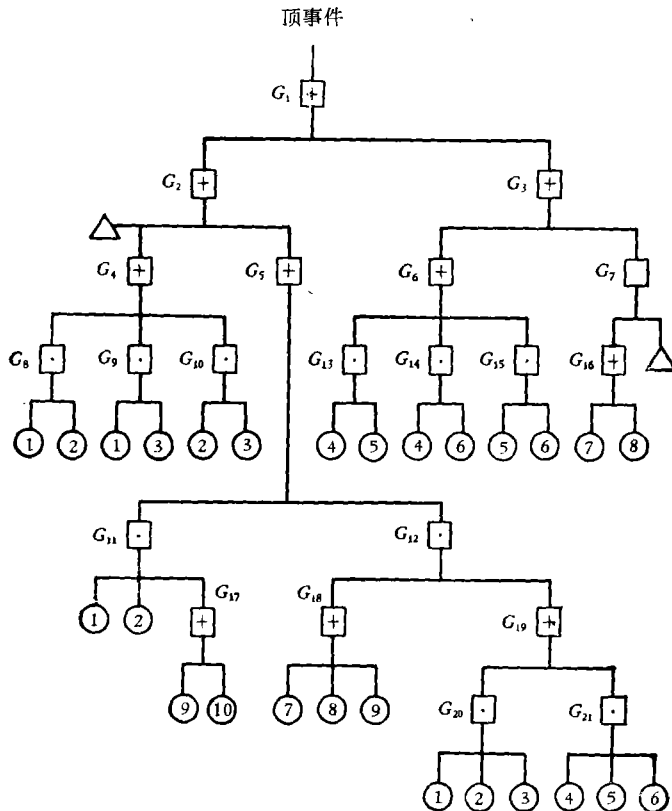


图 2

为有用。后者数量大, 罗列出来不易抓重点。前者是其“分组代表”, 数量集中, 便于掌握。需要时由“模块最小割集”, 可立即写出“事件最小割集”。

五、举 例

例 1. 我们对比地应用新、老途径分析图 1 失效树, 为此详细列出计算步骤以比较计算量。

先按一般途径分析。第一步枚举最小割集

$$\begin{aligned} G_0 &= G_1 G_2 = AG_3(B + E + G_4) = A(B + H)[B + E + D(C + F)] \\ &= ABB + ABE + ABCD + ABDF + AHB + AHE + AHDC + AHDF \\ &= AB + AHE + AHDC + AHDF. \end{aligned}$$

第二步 化不交型

1) 对 AB 化

$$\begin{aligned} \overline{AB}(AHE + AHDC + AHDF) &= (\bar{A} + \bar{B})(AHE + AHDC + AHDF) \\ &= \bar{B}AHE + \bar{B}AHDC + \bar{B}AHDF. \end{aligned}$$

2) 对 $\bar{B}AHE$ 化

$$\begin{aligned} \overline{\bar{B}AHE}(\bar{B}AHDC + \bar{B}AHDF) &= (B + \bar{A} + \bar{H} + \bar{E})(\bar{B}AHDC + \bar{B}AHDF) \\ &= \bar{E}\bar{B}AHDC + \bar{E}\bar{B}AHDF. \end{aligned}$$

3) 对 $\bar{E}\bar{B}AHDC$ 化

$$\begin{aligned} \overline{\bar{E}\bar{B}AHDC}(\bar{E}\bar{B}AHDF) &= (E + B + \bar{A} + \bar{H} + \bar{D} + \bar{C})\bar{E}\bar{B}AHDF \\ &= \bar{C}\bar{E}\bar{B}AHDF. \end{aligned}$$

故 $\phi = AB + \bar{B}AHE + \bar{E}\bar{B}AHDC + \bar{C}\bar{E}\bar{B}AHDF$.

下面按新途径来分析:

$$\begin{aligned} \phi &= G_0 = G_1 G_2 = AG_3(B + \bar{B}E + \bar{B}\bar{E}G_4) \\ &= A(B + \bar{B}H)[B + \bar{B}E + \bar{B}\bar{E}D(C + \bar{C}F)] \\ &= ABB + AB\bar{B}E + AB\bar{B}\bar{E}DC + AB\bar{B}\bar{E}D\bar{C}F + A\bar{B}HB \\ &\quad + A\bar{B}H\bar{B}E + A\bar{B}H\bar{B}\bar{E}DC + A\bar{B}H\bar{B}\bar{E}D\bar{C}F \\ &= AB + A\bar{B}HE + A\bar{B}H\bar{E}DC + A\bar{B}H\bar{E}D\bar{C}F. \end{aligned}$$

结果相同, 按新途径大大省去计算量, 而且可用单一程序来完成。

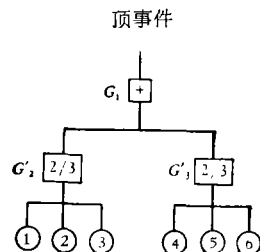


图 3

新途径再结合采用分解简约办法,计算量还将大幅度减少,例 3 将综合表明这个效果。

例 2. 取自文献 [5]. 失效树如图 2.

首先进行简约. 可以看出,除 G_4, G_6 以下子树各自代表三中取二的表决关系外, G_5, G_7 以下子树都是逻辑多余的. 例如 G_{11} 比 G_8 为多余, G_{12} 比 G_4 或 G_6 为多余, G_7 比 G_4 为多余. 因此我们把图 2 失效树简约成图 3.

再运用不交型运算规则,即可写出不交型失效函数

$$\begin{aligned} DFF = G_1 = G'_2 + \overline{G'_2 G'_3} &= (x_1 x_2 + \bar{x}_1 x_2 x_3 + \bar{x}_2 x_3 x_1) \\ &+ (\bar{x}_1 \bar{x}_2 + x_1 \bar{x}_2 \bar{x}_3 + x_2 \bar{x}_3 \bar{x}_1)(x_4 x_5 + \bar{x}_4 x_5 x_6 + \bar{x}_5 x_6 x_4) \\ &= x_1 x_2 + \bar{x}_1 x_2 x_3 + \bar{x}_2 x_3 x_1 + \bar{x}_1 \bar{x}_2 x_4 x_5 + \bar{x}_1 \bar{x}_2 \bar{x}_4 x_5 x_6 + \bar{x}_1 \bar{x}_2 \bar{x}_5 x_6 x_4 \\ &+ x_1 \bar{x}_2 \bar{x}_3 x_4 x_5 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 x_5 x_6 + x_1 \bar{x}_2 \bar{x}_3 \bar{x}_5 x_6 x_4 + x_2 \bar{x}_3 \bar{x}_1 x_4 x_5 \\ &+ x_2 \bar{x}_3 \bar{x}_1 \bar{x}_4 x_5 x_6 + x_2 \bar{x}_3 \bar{x}_1 \bar{x}_5 x_6 x_4. \end{aligned}$$

例 3. 失效树如图 4, 取自文献 [6].

第一步 分解简约.

本例特点是重复事件很多 (25 种), 存在着重复模块如 G_{13} 和 G_{26}, G_8 和 G_{28} . 我们还可用割顶点办法找出一些重复模块, 例如从 G_5, G_{15} 割出 G'_5, G'_{15} , 各以 $x_{14}, x_{17}, x_{18}, x_{22}$ 为输入, 即是复重模块. 这样得到重复模块如表 2.

表 2

重复模块符号	在失效树中位置	输入底事件
a	G_{13}, G_{26}	$x_{11}, x_{25}, x_{10}, x_{24}$
b	G_8, G_{28}	x_9, x_{23}
c	$G_{16}, G'_{33}, G'_{23}, G_{30}$	x_{12}, x_3
d	$G'_{17}, G_{24}, G_{34}, G'_{35}$	x_2, x_4
e	G'_7, G'_{27}	x_5, x_8, x_{19}
f	G'_{12}, G'_{25}	x_6, x_{12}, x_{20}
g	G'_5, G'_{15}	$x_{14}, x_{17}, x_{18}, x_{22}$
h	G'_2, G'_{14}	x_{13}, x_{10}, x_{21}

这些重复模块各用“准底事件”代替, 并计入剩余底事件 x_7, x_{15} , 按指数律估计, 可把计算量减少

$$2^{25} - 2^{(8+2)} = 2^{15} \text{ 倍.}$$

如不用复重模块概念, 本例是无法分解简约的.

第二步. 用不交型运算规则直接写出不交型失效函数, 并展开和吸收归并, 得

$$\begin{aligned} F = & hg + h\bar{g}af(x_{15} + \bar{x}_{15}be + \bar{x}_{15}\bar{b}\bar{e}cx_7 + \bar{x}_{15}\bar{b}\bar{e}c\bar{x}_7d) + h\bar{g}a\bar{f}dx_7(x_{15} + \bar{x}_{15}be + \bar{x}_{15}\bar{b}\bar{e}c) \\ & + h\bar{g}a\bar{f}d\bar{x}_7c(x_{15} + \bar{x}_{15}b) + \bar{h}b\bar{e}g(x_{15} + \bar{x}_{15}af + \bar{x}_{15}a\bar{f}dx_7 + \bar{x}_{15}a\bar{f}d\bar{x}_7c) \\ & + \bar{h}b\bar{e}\bar{g}(af + a\bar{f}dx_7) + \bar{h}b\bar{e}\bar{g}a\bar{f}d\bar{x}_7c + \bar{h}b\bar{e}cx_7g(x_{15} + \bar{x}_{15}af + \bar{x}_{15}a\bar{f}d) \\ & + \bar{h}b\bar{e}cx_7\bar{g}(af + a\bar{f}d) + \bar{h}b\bar{e}c\bar{x}_7dg(x_{15} + \bar{x}_{15}a) + \bar{h}b\bar{e}c\bar{x}_7d\bar{g}a, \end{aligned}$$

其中

$$\begin{aligned} a &= (x_{11} + \bar{x}_{11}x_{25})(x_{10} + \bar{x}_{10}x_{24}), & e &= x_5 + \bar{x}_5x_8 + \bar{x}_5\bar{x}_8x_{19}, \\ b &= x_9 + \bar{x}_9x_{23}, & f &= x_6 + \bar{x}_6x_{12} + \bar{x}_6\bar{x}_{12}x_{20}, \end{aligned}$$

集”。从 $hafx_{15}$ 到 $bcdgx_{15}$ 共五个模块最小割集, 因为 a 是“与”门, 总是代表二个事件。所以写成五阶“事件最小割集” 188 个。以上和文献 [6] 求得的最小割集数完全一致。另外, 我们还求得在文献 [6] 中用大型计算机没有求出的六阶最小割集 168 个, 即“模块最小割集” $hadx_7x_{15}$, $hadcx_{15}$, $beadx_7$, bcx_7af 所代表的。这样, 我们求得 11 个“模块最小割集”实际上包括了全部“事件最小割集” 392 个。并且判定: 没有七阶及七阶以上的最小割集存在。所以求得的 $P(F)$ 是准确值。

作者衷心感谢杨嘉墀、屠善澄、桂湘云、何国伟、曹晋华、程侃、黄锡滋、梅启智、黄祥瑞、赵子厚等同志的审阅、讨论和帮助。

附 录

我们利用 Venn 图对 Boolean 代数的一般运算规则和不交型运算规则作一直观比较。

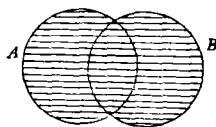
1) “交”、“补”运算规则相同。

2) “并”运算规则的形式不同。

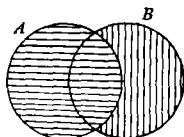
为符号书写方便起见, 用 $+$ 代 $\cup$, 用 $\cdot$ 代 $\cap$, $\cdot$ 常省略。

在 Boolean 代数中, $A + B = B + A$ 相应于附录图 1。在不交型 Boolean 代数中, $A + \bar{A}B$, $B + \bar{B}A$ 分别相应于附录图 2, 3。而且 $A + \bar{A}B = B + \bar{B}A$ 。

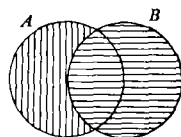
比较可见, 附录图 1—3 实质上是一致的。



附录图 1



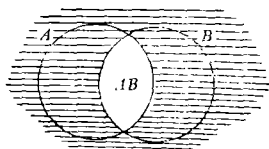
附录图 2



附录图 3

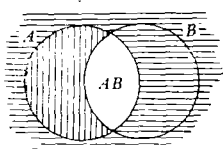
并 $A + B$ 似乎简单, 但由于暗含了“面积”(粗浅地说) $S_A + S_B - S_{AB}$ 的关系, 将给以后的计算带来麻烦。而不交型 $A + \bar{A}B$ 或 $B + \bar{B}A$, 从根本上消除了这种麻烦, 因而可能取得简化的效果。以下二条同此讨论。

3) DeMorgan 律和不交型 DeMorgan 律的比较。



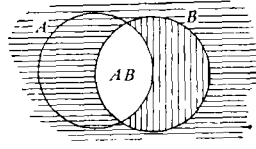
$$\overline{AB} = \bar{A} + \bar{B}$$

附录图 4



$$\bar{A}B = \bar{A} + A\bar{B}$$

附录图 5



$$\overline{AB} = \bar{B} + B\bar{A}$$

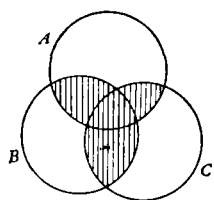
附录图 6

比较可见, 附录图 4—6 在实质上是一致的。

$\overline{A + B} = \bar{A}\bar{B}$ 与 $\overline{A + \bar{A}B} = \bar{A}\bar{B} = \overline{B + \bar{B}A}$ 可同理比较。

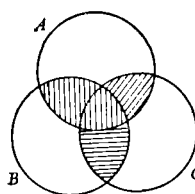
4) 三中取二表决门运算规则的比较

比较可见, 附录图 7, 图 8 实质上是一致的。



$$z = AB + BC + CA$$

附录图 7



$$z = AB + \bar{A}BC + \bar{B}CA$$

附录图 8

$$\left(\begin{array}{l} z = BC + \bar{B}AC + \bar{C}AB \\ z = CA + \bar{C}AB + \bar{A}BC \end{array} \right. \text{可同理比较})$$

参 考 文 献

- [1] Cargal, J. M., *IEEE Trans. on Reliability*, V. R-29(1980), 269—272.
- [2] Rosenthal, A., in *Reliability and Fault-Tree Analysis* (Ed. Barlow, R. E., Fussell, J. B. & Singpuwala, N. D.), the Society of Industrial and Applied Mathematics (SIAM), 1975, 135—152.
- [3] Birnbaum, Z. W. & Esary, J. D., *J. of the Society for Industrial and Applied Mathematics*, 15 (1965), 444—468.
- [4] Bennetts, R. G., *IEEE Trans. on Reliability*, V. R-24(1975), 175—185.
- [5] Vesely, W. Z., *Nuclear Engineering and Design* (An international journal), 13(1970), 337—360.
- [6] Camarda, P., et al., *IEEE Trans. on Reliability*, V. R-27(1978), 215—221.