文章编号:1671-4598(2023)07-0265-06 DOI:10.16526/j.cnki.11—4762/tp.2023.07.038 中图分类号:TP3 文献标识码:A

基于全局最优的信息启发式 Rollout 测试序列生成算法

王晓明[1], 袁乾臣[1], 王婧舒[1], 李 璠[2]
(1. 北京宇航系统工程研究所, 北京 100076; 2. 北京质远恒峰科技有限公司, 北京 100080)

摘要: 为实现电子设备或电气系统快速故障诊断, 在故障诊断过程中需生成测试序列, 一般通过分析测试点对故障的检测与隔离的次序能够得出产品测试序列生成排故引导树; 然而针对不同故障模式的故障率不同, 不同测试点的测试权重、测试费用、测试时间不同, 从不同测试点出发形成的排故引导树也不同; 提出了一种基于 A0 * 信息启发式 Rollout 策略的测试序列生成算法, 其作为一种近优的结算方法既考虑可靠性, 也考虑测试费用最小, 既减轻庞大的计算量, 又获得了比次优启发式算法更好的诊断结果。

关键词: 故障诊断; 排故引导树; Rollout 策略; 测试序列

Generation Method for Information Heuristic Rollout Test Sequence Based on Global Optimization

WANG Xiaoming[1], YUAN Qianchen[1], WANG Jingshu[1], LI Fan[2]
(1. Beijing Institute of Aerospace Systems Engineering, Beijing 100076, China;
2. Beijing Zhiyuan Hengfeng Co., Ltd., Beijing 100080, China)

Abstract: In order to realize the rapid fault diagnosis of electronic devices or electrical systems, test sequence needs to be generated in the fault diagnosis. Generally, through the analysis of the order of fault detection and isolation for test points, the fault-guided tree generated from the product testing sequence can be obtained. However, for the different failure rates of different failure modes, the test weight, cost and time of different test points are different, and the fault-guided trees formed from different test points are also different. A test sequence generation algorithm based on AO * information heuristic Rollout strategy is proposed, which is a near-optimal settlement method, the method considers both reliability and minimum testing costs, which not only reduces massive computation, but also obtains better diagnostic results than the suboptimal heuristic algorithm.

Keywords: fault diagnoses; fault-guided tree; Rollout strategy; test sequence

0 引言

相关性模型 (dependency model), 也称为相关性矩阵, 是测试性建模的经典模型[1-3]。相关性模型是对电气产品的组成、故障模式、故障率、测试点、测试方法以及它们之间的逻辑关系进行描述的模型, 其数学表达为依存矩阵 (dependency matrix, 简称为 D 矩阵)[2]。以 D 矩阵为输入, 通过分析测试点对故障的检测与隔离的次序能够得出产品测试序列即排故引导树。然而故障模式的故障率不同, 不同测试点的测试权重、测试费用、测试时间都是不同, 从不同测试点出发所形成的排故引导树也是不同[4]。

在解算复杂电气产品的相关性矩阵时往往消耗的时间巨大[5]。目前, 常用的诊断策略构建方法主要包括: 目前解决诊断策略设计问题的方法主要有三类: DP 算法、群智能算法和启发式搜索算法。

1) DP 算法:

DP 是一种递归算法, 其中故障诊断树形成过程是由上而下, 按通过优先方法进行搜索, 隔离出全部故障为止[17]。对于诊断策略优化问题, DP 算法的储存与计算需求为, 其中 m 为故障数, n 为测试数量, 因此当 n 较小时是可行的, 而对于大的复杂系统, 其储存和计算量将呈指数增长。因而不适合复杂电气产品的诊断策略设计工作。

2) 群智能算法:

群智能算法简单来说就是一种一类仿生算法, 仿造自然界中某些规律进行优化, 常见的有蚁群算法、粒子群算法等。

蚁群算法是一种用来解决多线路最优的概率型算法。Dorigo 等[15-16]根据蚂蚁寻找食物过程中路径寻优的行为提

收稿日期:2023-05-26; **修回日期**:2023-05-31。

作者简介:王晓明(1983-),女,山东昌乐人,硕士研究生,高级工程师,主要从事复杂系统的测试性故障诊断与启发式算法优化方向的研究。

引用格式:王晓明,袁乾臣,王婧舒,等.基于全局最优的信息启发式 Rollout 测试序列生成算法[J].计算机测量与控制,2023,31(7):265-270.

出了蚁群算法。目前最常用的解决方式是将测试序贯优化问题转换为搜索最小完备测试序列问题，进而利用蚁群的记忆性与信息素积累反馈机制解决该问题。该算法着眼于每一只“蚂蚁”的搜索，结构简单编程容易，算法具备反馈机制，可通过反馈不断修正缺陷。但测试序列优化问题与搜索最小完备测试序列问题并不完全等同，得到的测试序列与诊断策略存在区别，应用受限。

3）启发式搜索算法：

常用于诊断策略优化设计问题的启发式搜索算法有贪婪算法、AO＊算法、准深度算法等。

贪婪算法是快速搜索算法。该算法搜索速度快，采用固定顺序方法构造诊断树。该算法对整个诊断树采取局部择优搜索的策略，搜索速度极快，但效果不佳。

AO＊算法被广泛应用于测试性设计中，如 TEAMS 软件就采用 AO＊生成诊断树。算法优势是可以找到近似的全局最优解，相比于 DP 算法效率有所提高，但需要处理的数据非常大，计算复杂，不适合大型系统。

准深度算法可以看成是深度算法的简化，类似于贪婪算法，但考虑更多，每一步考虑的是之后的诊断树的预估值，因而速度较快，大概率可找到全局最优解。但该算法结构复杂，且缺少反馈过程，难以应用于拓展场合。

以上三种启发式搜索算法各有优劣，在解决不可靠测试条件下的诊断策略优化问题过程中可发挥一定作用。

本项目提出了一种结合全局最优的启发式 AO＊算法的 Rollout 策略的排故引导策略生成方法，其作为一种近优的结算方法在解的复杂度方面要低于启发式搜索算法，能够确保运算速率。

1 基于全局最优的启发式 AO＊算法的测试序列生成方法

1.1 AO＊算法概述

测试序列的生成方法直接影响故障诊断的准确度和执行效率，基于全局最优的启发式 AO＊算法所生成的测试序列是一种全局最优的测试策略[6]，该算法同三种信息启发式（霍夫曼编码、熵、熵＋1）函数相结合，通过向下扩展以及向上反馈修正两个基本操作来得到最优的测试序列，它的计算量小于 DP 算法和群智能算法，在搜索过程中会进行不断回溯，以实现最优解[7]。

1.2 AO＊算法的基本元素及输出策略

AO＊算法的输入包括以下内容：

1）系统状态集合：

$$S=\{s_0,s_1,\ldots s_m\} \tag{1}$$

其中：s_0 为系统的无故障状态，s_i 为故障状态 1～m。

2）故障概率向量：

$$p=[p(s_0),p(s_1),\ldots p(s_m)]^T \tag{2}$$

其中：$p(s_0)$ 为系统完好率，$p(s_1),\ldots p(s_m)$ 为各故障模式的频数比＊（1－完好率）。

3）测试集合：

$$t=\{t_1,t_2,\ldots t_n\} \tag{3}$$

4）成本向量（时间、人力、其他经济因素）：

$$c=[c_1,c_2,\ldots c_n]^T \tag{4}$$

AO＊算法的输出综合考虑故障发生概率、测试时间，测试费用等的最优诊断树，AO＊算法使用启发式评估函数（HEF）指导测试搜索过程[8]，其计算公式为：

$$w^*(x)=[\hat{p}(x)]^{-1}\sum_{f_i\in x}\omega_i(x)p(f_i) \tag{5}$$

$$h(x)=HEF_1=\sum_{j=1}^{w'(x)}c_j+[w^*(x)-w'(x)]\cdot c_{w'(x)+1} \tag{6}$$

其中：所有故障模式按照故障概率从大到小排列所得为 $p(f_i)$，所有测试按照成本从小到大排列所得为 c_j。x 为系统所处状态可能包含的故障集，$\hat{p}(x)$ 为该故障集的概率和。w_i 是集合中元素 f_i 的 Huffman 编码字长度。$w'(x)$ 为 $w^*(x)$ 的整数部分。

1.3 AO＊算法步骤

采用 AO＊算法实现测试序列的优化的基础是启发式评估函数的构造。AO＊算法是基于启发式评估函数 $h(x)$，选择最有可能达到目标节点的子节点进行扩展。其基本步骤如下。

Step 1：建立一个搜索图 G，使其仅仅包含起始节点 S，设 $F(s)=h(s)$。如果 S 为终节点，则标记 S 为 SOLVED，离开算法步骤。

Step 2：重复以下步骤，直到 S 已经标记为 SOLVED，此时 $J=F(s)$ 为期望的测试代价，并以标记的解树为测试算法。

Step 2.1：通过跟踪 G 中从 S 出发的、有标记的、连接符，计算 G 中的一个局部解图 G'。选择 G' 中最高 $h(x)$ 的节点 x 进行扩展。最初时 $x=S$（x 为系统状态集合）。

Step 2.2：扩展节点 x，生成它的所有后继节点的二元集合，表示为 $(x_{jp},x_{jf}),t_j\notin \Pi x$。其中，$\Pi x$ 为在通向 x 的路径上已经标记使用的测试集。初始 Πx 为空。对于每一个在 G 中未曾出现过的 x 的后继节点，设：

$$F(y)=h(y) \tag{7}$$

$$y=x_{jp},x_{jf},t_j\notin \Pi x \tag{8}$$

其中：如果任意 y 属于终节点，则标记其为 SOLVED。

Step 2.3：建立一个仅包含节点 x 的节点集合 Z。

Step 2.4：执行以下步骤，直到 Z 为空：

Step 2.4.1：从 Z 中移出这样的节点 y，这个 y 在 G 中的后裔不出现在 Z 中。（移出当前没有后裔的，最底层的节点进行分析）

Step 2.4.2：修正 y 的值如下所示：

$$e=\min_{t_j\in \Pi y}\{c_j+p(y_{jp})F(y_{jp})+p(y_{jf})F(y_{jf})\} \tag{9}$$

其中：$p(y_{ip})$ 为相对概率。令 k 为测试代价最小值的坐标，并对这个具有最小值的连接符加以标记，包括 (y,y_{kp}) 和 (y,y_{kf})。如果 y_{kp} 和 y_{kf} 都已标记 SOLVED，则标记此节点 y 为 SOLVED。

Step 2.4.3：如果 $F(y)\neq e$，设 $F(y)=e$。（由于测试的选择及向下扩展的节点而导致的成本改变，修正 y 的

值为选择当前最小成本的测试 t_k 的情况下的成本值 e。

Step 2.4.4：如果 $F(y)$ 在 2.4.3 中修正了值，或已标记为 SOLVED，则把沿着标记路径的 y 的所有父辈节点都添加到 Z 中，忽略了没有由标记的连接符连接到 y 的先辈节点。(若 y 被修正或已下溯到根节点，则将其上游的、当前标记路径中的先辈节点都添加到 Z 中，对先辈节点进行修正计算，即执行 2.4.1 到 2.4.4 的步骤，直到 Z 为空。(Z 为空表示已回溯结束，或已到起始节点，或已到没有发生修正的节点)。

2　基于 Rollout 策略的测试序列生成方法

2.1　Rollout 算法概述

Rollout 算法的基本思想是用一个基准策略经过 Rollout 仿真得到一个更新策略，再把更新策略作为基准策略进行迭代更新，逐步逼近最优策略[9]。该方法能得到比基准策略更加精确的结果，但是不能保证是全局最优解。

2.2　Rollout 算法基本元素

设某产品共有 $(m+1)$ 类状态，故障测试相关矩阵表示为 $D=[d_{ij}]$，根据测试结果值的不同，可分为二值测试和多值测试，二值测试即 d_{ij} 只有两个状态，比如 (0，1)、(通过，不通过)，多值测试即 d_{ij} 存在三个或三个以上的状态[9]，比如 (0，0.5，1)、(0，1，2，3)、(通过，不通过，待定)。

测试序列产生目标是合理规划测试顺序，使得在达到故障隔离目的的情况下测试费用最小。

定义所需测试费用为：

$$J=\sum_{i=0}^{m}\left\{\sum_{j=0}^{|P_i|}c_{i[j]}\right\}\cdot p(s_i) \tag{10}$$

其中：p_i 表示隔离故障 s_i 所用的测试集合，$|p_i|$ 表示测试集合的容量。

2.3　Rollout 算法步骤

Step 1：初始化创建一个模糊集 Z，只包含根节点 S。初始化一个图 G。如果 S 是终端节点，则将 S 加入到 G，则 G 就是问题的解。

Step 2：重复以下操作，直到模糊集 Z 为空后停止，则 G 就是问题的解。

Step 2.1：从模糊集 Z 中移除一个模糊集节点 x_i，加入到图 G 中，如果 x_i 是一个已解节点则从模糊集 Z 中取下一个模糊集节点。对模糊节点 x_i 的可用测试集 T_j 中的所有测试 t_j，重复以下步骤。

Step 2.1.1：初始化一个模糊集节点 Y'，创建一个图 G'，将测试 t_j 加入到图 G' 中，用 t_j 将 x_i 分解成两个模糊子集，测试通过 x_{ijp} 和测试失败 x_{ijf}，将 x_{ijp} 和 x_{ijf} 加入到 Y' 中，重复以下步骤直到 Y' 为空。

Step 2.1.1.1：从 Y' 中移除一个模糊集节点 y'_r，加入到图 G' 中，如果 y'_r 是一个已解节点，则选择下一个模糊集节点。计算其可用测试集 T'_r 中每个测试的单位测试费用的信息增益 $b(y'_r,t'_l)$，$t'_l\in T'_r$；采用单位费用信息熵的方法来计算信息增益：

$$b(x,t_j)=\frac{IG(x,t_j)}{c_j} \tag{11}$$

$$IG(x,t_j)=-p(x_{jp})\log_2 p(x_{jp})-p(x_{jf})\log_2 p(x_{jf}) \tag{12}$$

Step 2.1.1.2：选择信息增益最大的测试 $t'_{\max}$，

$$b(y'_r,t'_{\max})=\max_{t'_l\in T'_r}b(y'_r,t'_l) \tag{13}$$

将 $t'_{\max}$ 加入到图 G' 中，并用 $t'_{\max}$ 将 y'_r 分成两个模糊子集，加入到 Y' 中。

Step 2.1.2：计算测试的费用开销：

$$h(x_{ij},)=\sum_{q=1}^{l_j}\left\{\left(\sum_{r=1}^{|P_q|}c_{P_q[r]}\right)\cdot\frac{p(x_l)}{p(x_{ijp})}\right\} \tag{14}$$

其中：x_l 是 x_{ijp} 的模糊子集，l_j 表示 x_{ijp} 的模糊子集个数，P_q 表示隔离的节点到 x_{ijp} 节点的测试序列，$|P_q|$ 表示测试序列中的元素个数，$c_{P_q[r]}$ 表示 P_q 的第 r 个测试费用。

Step 2.2：令

$$h_{t_j}(x_i)=c_j+h(x_{ijp})p(x_{ijp})+h(x_{ijf})p(x_{ijf}) \tag{14}$$

选择 $t^*\in T_j$ 使得：

$$h(x_i,t^*)=\min_{t_j\in T_j}h_{t_j}(x_i) \tag{15}$$

并将节点 x_i 展开。将 t^* 加入到图 G 中。把用 t^* 拆解的模糊子集 x_{ip} 和 x_{if} 加入 Z 中，然后返回 Step 2.1。

3　基于全局最优的 A0 * 信息启发式 Rollout 算法

为了减轻计算量并获得比启发式算法更好的结果，我们将 Rollout 策略与基于全局最优的 AO * 启发式方法相结合，以基于错误状态概率，测试成本和依赖矩阵有效地构建测试序列。

3.1　算法概述

此算法中，如果它最大化了以下测试的单位成本信息增益，则从歧义状态中选择测试。

$$k=\arg\max_j\left\{\frac{IG(x,t_j)}{c_j}\right\} \tag{16}$$

其中：$IG(x,t_j)$ 是信息增益，由以下公式给出：

$$IG(x,t_j)=-\{p(x_{jp})\log_2 p(x_{jp})\log_2 p(x_{jp})+p(x_{jf})\log_2 p(x_{jf})\log_2 p(x_{jf})\} \tag{17}$$

因此，信息启发式是一个具有计算复杂性 $O(mn)$ 的一步前向程序。其它相关启发式有“分别启发式”，$d_c(x,t_j)$，(又称为可区分性启发式)，定义如下：

$$d_c(x,t_j)=p(x_{jp})\cdot p(x_{jf}) \tag{18}$$

选择一个测试 t_k，以最大化每个歧义节点的可区分性标准。很容易表明，测试成本相等时，信息启发式提供了与可分辨性标准相同的测试树。基于多义的或节点对决策树进行的深度优先扩展，信息启发式 (称为多步信息启发式算法) 的变体涉及给定级别 (基于多步前向) 的选择测试。测试给定级别的有效性仍根据其每单位测试成本的信息增益进行评估。

3.2　算法步骤

Step 1：建立一个节点集 Z，仅包含根节点 S 与一个空图形 G。如果 S 是一个终止节点，添加 S 至图形 G 中，退出解。

Step 2：重复下列步骤至 Z 为空集，然后退出，以解决方案树作为测试算法。

Step 2.1：从 Z 中移出一个或节点 i 至 G。如果 i 是一个目标节点，继续 Z 中的下一个或节点；否则，考虑可行测试集 T_i 中的每一个测试 t_i，它们将节点 i 分为通过/失败子集：x_{ijp} 与 x_{ijf}（i 的直接后续或节点）。创建一个集合 Y 与图形 G'，两者均由 i 的直接后续或节点构成。重复下列程序（信息启发式）直至 Y 为空。

Step 2.1.1：从 Y 中移除一个或节点 r。如果这是一个非目标节点，计算该节点每一个可行测试的单位成本信息增益；否则，前往 Y 的下一个或节点。

Step 2.1.2：选择 r 中单位成本信息增益达到最高的测试。通过选择最小指数的测试解决关系。将该项测试加入 G'。分别根据选定测试的通过 / 失败结果将 r 分为通过/失败子集。将获得的通过/失败或节点加入 Y 与 G'。

Step 2.2：基于测试序列（存于 G'），计算每个 i 的后续或节点预期测试成本（如 x_{ijp}），公式如下：

$$h(x_{ijp}) = \sum_{i=0}^{|\overline{x}|} \left\{ \sum_{k=1}^{|P_i|} c_{P_i[k]} \right\} p(\overline{x_i}) \tag{19}$$

其中：$\overline{x}$ 是目标节点集，或由节点 x_{ijp} 生成的测试树构建，而 $|\overline{x}|$ 表示 $\overline{x}$ 的基数。此处，P_i 表示应用于孤立节点集 x_{ijp} 系统状态 $\overline{x}$ 的测试序列，$|P_i|$ 是它的基数。

Step 2.3：计算或节点 i 的每个候选测试 $t_i \in T_i$ 的期望测试成本，公式如下：

$$h_{t_i}(x_i) = c_j + p(x_{ijp})h(x_{ijp}) + p(x_{ijf})h(x_{ijf}) \tag{20}$$

其中：$h(x_{ijp})$ 和 $h(x_{ijf})$ 分别对应通过/失败或节点子集 x_{ijp} 与 x_{ijf} 生成测试树中的预期测试成本。

Step 2.4：选择 $t^* \in T_i$，获得最小期望测试成本。使用 Step 2.1 信息启发式，通过有利于测试生成，还有最小指数目标解决关系。将 t^* 加入 G，然后分别将 t^* 测试生成的通过/失败结果加入通过/失败或子集 G 与 Z。

3.3 算法实例

表 1 给出某电气系统的相关性矩阵[10-12]。$S(i)$ 为某电气系统的故障模式，TPj 为系统的测试项，$P(si)$ 为 $S(i)$ 的故障模式频数比。在相关性矩阵中 1 代表测试项 TP 与故障模式 S 相关，0 代表不相关。

表 1 某电气系统相关性矩阵

$S(i)$	TPj					$p(si)$
	t_1	t_2	t_3	t_4	t_5	
	成本向量 c_j					
	1	1	1	1	1	
S_0	0	0	0	0	0	0.70
S_1	0	1	0	0	1	0.01
S_2	0	0	1	1	0	0.02
S_3	1	0	0	1	1	0.10
S_4	1	1	0	0	0	0.05
S_5	1	1	1	1	0	0.12

初始节点 S 包含所有可能状态 $s_0, s_1, \ldots s_5$。首先，我们采用 Step 2.1，应用图 1 列举的所有 5 种可行测试将 S 分为通过/失败节点。

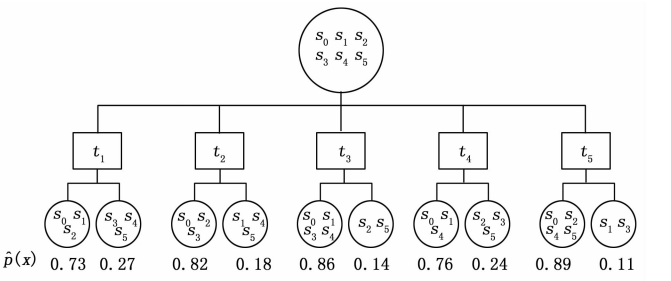

图 1 测试分割

对于 t_1 子树，分别有子集 $\{s_0, s_1, s_2\}$ 与 $\{s_3, s_4, s_5\}$ 对应通过与失败或节点。我们在 $\{s_0, s_1, s_2\}$ 上执行步骤 Step 2.1.1 和 Step 2.1.2（即信息启发式）。

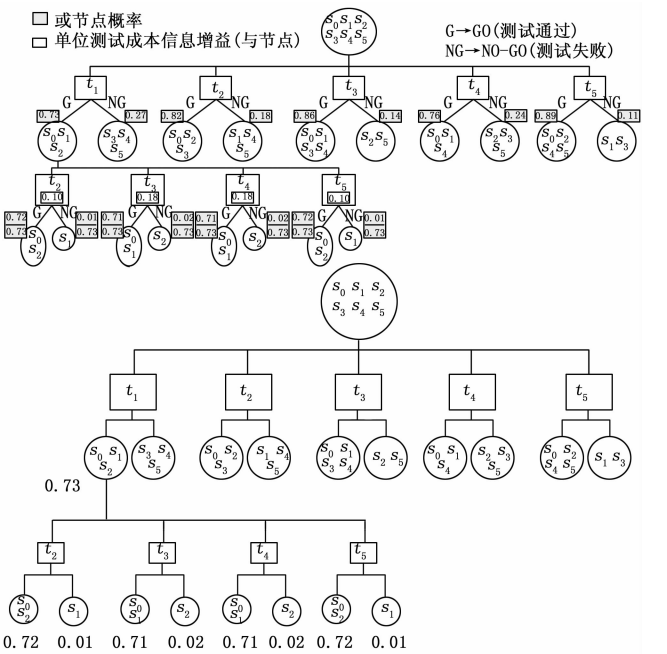

图 2 测试 t_1 通过选择第二个测试点

测试 t_2 单位成本信息增益：

$$IGIG(x,t_2) = -\{p(x_{2p})\log_2 p(x_{2p})\log_2 p(x_{2p}) + p(x_{2f})\log_2 p(x_{2f})\log_2 p(x_{2f})\} = -\frac{0.72}{0.73} * \log_2 \frac{0.72}{0.73} + \frac{0.01}{0.73} * \log_2 \frac{0.01}{0.73} = 0.105 \tag{21}$$

测试 t_3 单位成本信息增益：

$$IG(x,t_3) = -\{p(x_{3p})\log_2 p(x_{3p})\log_2 p(x_{3p}) + p(x_{3f})\log_2 p(x_{3f})\log_2 p(x_{3f})\} = -\frac{0.71}{0.73} * \log_2 \frac{0.71}{0.73} + \frac{0.02}{0.73} * \log_2 \frac{0.02}{0.73} = 0.181\ 2 \tag{22}$$

测试 t_4 单位成本信息增益：

$$IG(x,t_4) = -\{p(x_{4p})\log_2 p(x_{4p})\log_2 p(x_{4p}) + p(x_{4f})\log_2 p(x_{4f})\log_2 p(x_{4f})\} = -\frac{0.71}{0.73} * \log_2 \frac{0.71}{0.73} + \frac{0.02}{0.73} * \log_2 \frac{0.02}{0.73} = 0.181\ 2 \tag{23}$$

测试 t_5 单位成本信息增益：

$$IG(x,t_5) = -\{p(x_{5p})\log_2 p(x_{5p})\log_2 p(x_{5p}) +$$

$$p(x_{5f})\log_2 p(x_{5f})\log_2 p(x_{5f})\}=$$

$$-\frac{0.72}{0.73}*\log_2\frac{0.72}{0.73}+\frac{0.01}{0.73}*\log_2\frac{0.01}{0.73}=0.105 \quad (24)$$

根据：

$$k=\arg\max_j\left\{\frac{IG(x,t_j)}{c_j}\right\} \quad (25)$$

得到 $k=\{3,4\}$。

t_2 和 t_5 的单位成本信息增益均为 0.104 4，t_4 与 t_3 的单位信息成本增益最大，为 0.181 2。t_4 与 t_3 的增益相等，选择测试序号较小的测试，则选择测试 t_3 为第二个测试点。

反复应用信息启发式，直到为或节点 $\{s_0,s_1,s_2\}$ 构建的测试子树完整，如图 3 所示。测试 t_2 和 t_5 分割结果相同，选择 t_2。

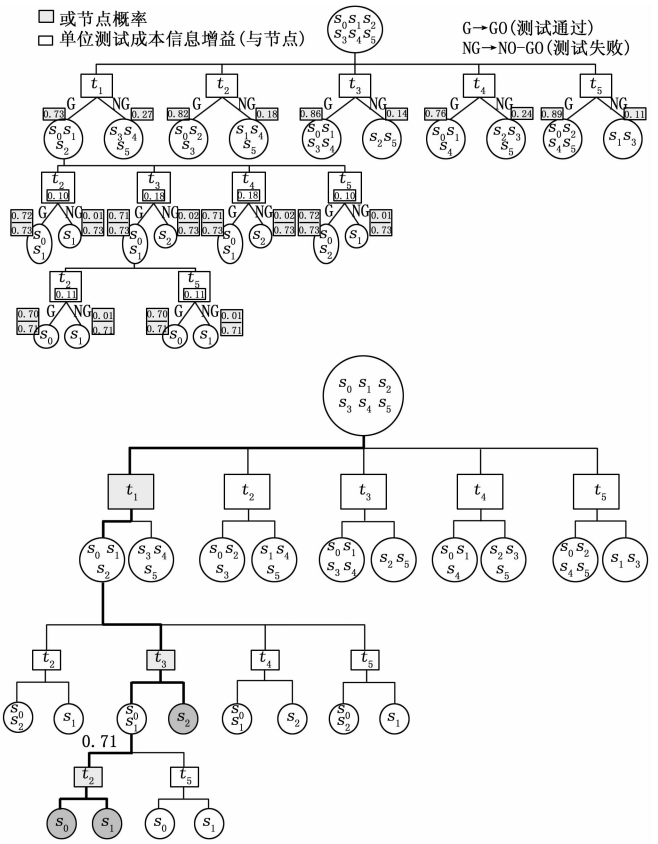

图 3　测试 t_1 通过侧生成完整诊断树

对测试 t_1 失败子集或节点 $\{s_3,s_4,s_5\}$ 执行相同的过程：我们在 $\{s_3,s_4,s_5\}$ 上执行 Step 2.1.1 和 Step 2.1.2（即信息启发式）。

测试 t_2 单位成本信息增益：

$$IG(x,t_2)=-\{p(x_{2p})\log_2 p(x_{2p})\log_2 p(x_{2p})+$$
$$p(x_{2f})\log_2 p(x_{2f})\log_2 p(x_{2f})\}=$$
$$-\frac{0.1}{0.27}*\log_2\frac{0.1}{0.27}+\frac{0.17}{0.27}*\log_2\frac{0.17}{0.27}=0.95 \quad (26)$$

测试 t_3 单位成本信息增益：

$$IG(x,t_3)=-\{p(x_{3p})\log_2 p(x_{3p})\log_2 p(x_{3p})+$$
$$p(x_{3f})\log_2 p(x_{3f})\log_2 p(x_{3f})\}=$$

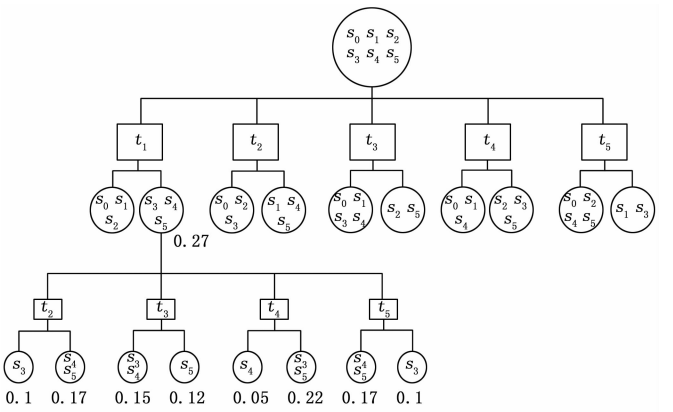

图 4　测试 t_1 不通过选择第二个测试点

$$-\frac{0.15}{0.27}*\log_2\frac{0.15}{0.27}+\frac{0.12}{0.27}*\log_2\frac{0.12}{0.27}=0.99 \quad (27)$$

测试 t_4 单位成本信息增益：

$$IG(x,t_4)=-\{p(x_{4p})\log_2 p(x_{4p})\log_2 p(x_{4p})+$$
$$p(x_{4f})\log_2 p(x_{4f})\log_2 p(x_{4f})\}=$$
$$-\frac{0.05}{0.27}*\log_2\frac{0.22}{0.27}+\frac{0.05}{0.27}*\log_2\frac{0.22}{0.27}=0.69 \quad (28)$$

测试 t_5 单位成本信息增益：

$$IG(x,t_5)=-\{p(x_{5p})\log_2 p(x_{5p})\log_2 p(x_{5p})+$$
$$p(x_{5f})\log_2 p(x_{5f})\log_2 p(x_{5f})\}=$$
$$-\frac{0.17}{0.27}*\log_2\frac{0.17}{0.27}+\frac{0.1}{0.27}*\log_2\frac{0.1}{0.27}=0.95 \quad (29)$$

综上所述，测试 t_3 单位成本信息增益最大，所以选择测试 t_3 为第二个测试点。

在 $\{s_3,s_4\}$ 上执行 Step 2.1.1 和 Step 2.1.2，计算得到，测试 t_2、测试 t_4、测试 t_5 单位成本信息增益相同，因此选择测试 t_2。

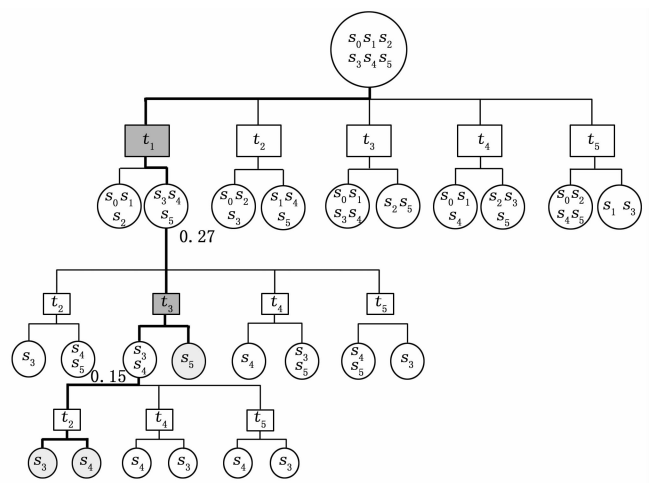

图 5　测试 t_1 不通过侧生成完整诊断树

最终采用 Rollout 信息启发式算法选择第一个测试 t_1 得到完整的诊断树如下：

可见，基于全局最优的 A0 * 信息启发式 Rollout 算法能够快速得到有效的测试序列树，基于测试序列树可进一

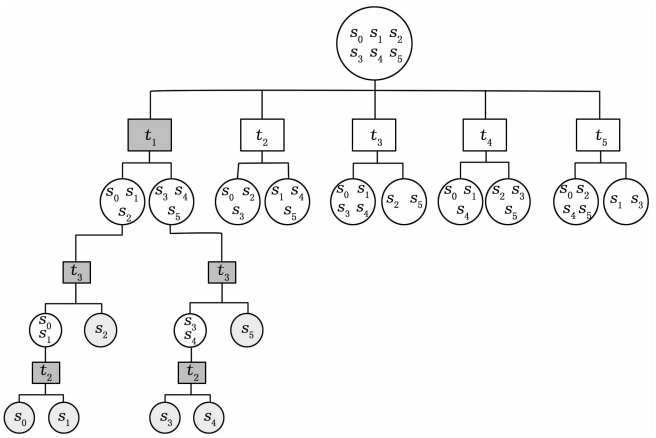

图 6 Rollout 信息启发式算法完整的诊断树

步转化为 ATE（自动测试设备）中 TPS（测试程序集）的测试程序，作为 ATE 的测试策略，同时，也可以进一步转化为符合 S1000D 标准和 GJB6600 的排故引导数据模块，作为 IETM（交互式电子技术手册）的重要组成部分。

4 结束语

装备的复杂度与性能一直呈正相关关系，这使得随着武器装备的发展，对装备开展测试性设计的难度也越来越大，需要考虑的因素也越来越多。如何利用现有的数据信息建立精度最高的测试性模型，采用何种算法才能快速有效地解决诊断策略生成问题，是测试性设计的核心问题。

采用基于全局最优的启发式 AO＊算法的测试序列生成方法可既考虑到可靠性，也考虑测试费用最小。其优点是诊断结果为“全局最优”诊断树。其缺点是由于在搜索过程中需要存储的临时支路和数据非常大[13]，一般只用于规模较小的系统[14]（故障模式＜50）。基于 Rollout 策略的测试序列生成方法既考虑可靠性，也考虑测试费用最小。其优点是诊断结果为“近似最优”诊断树，诊断效率高。既减轻庞大的计算量，又获得了比次优启发式算法更好的诊断结果。该方法缺点是需要算出全部潜在诊断树之后再进行寻优，不适用于规模过于庞大的系统。

参考文献：

[1] 国防科学技术工业委员会. GJB2547A-2012 装备测试性通用要求 [S]. 北京：国防科工委军标出版社，2012.

[2] 邱 静，刘冠军，杨 鹏，等. 装备测试性建模与设计技术 [M]. 北京：科学出版社，2012：297－305.

[3] PATTIPATI K R, ALEXANDRIDIS M G. Application ofheuristic search and information theory to sequentialfault diagnosis [J]. IEEE Trans on Systems, Man, andCybernetics, 1990, 20 (4)：872－887.

[4] 刘远宏，刘建敏，冯辅周，等. 基于 Rollout 信息启发式算法的故障诊断策略 [J]. 计算机工程，2015. 40 (8)：291－295.

[5] 杨 鹏，邱 静，刘冠军. 基于多值测试的诊断策略优化生成 [J]. 仪器仪表学报，2008，29 (8)：1675－1678.

[6] 黄以锋，景 博，罗炳海，等. 基于 Rollout 算法的序贯多故障诊断策略 [J]. 控制与决策，2015，30 (3)：572－576.

[7] ZHANG S G, HU Z, WEN X S. Test sequencing problem arising at the design stage for reducing life cycle cost [J]. Chinese J of Aeronautics, 2013, 26 (4)：1000－1007.

[8] TU F, PATTIPATI K R. Rollout Strategies for sequential fault diagnosis [J]. IEEE Trans on Systems, Man, and Cybernetics-Part A：Systems and humans, 2003, 33 (1)：86－99.

[9] 杨 鹏. 基于相关性模型的诊断策略优化设计技术 [D]. 长沙：国防科学技术大学，2008.

[10] 曾瑞林，王 冠，王潇宇，等. 火箭喷管极性自动化测试系统的改进设计研究 [J]. 宇航总体技术，2022：55－60.

[11] 马骏添，张素明，阎小涛，等. 基于图像识别的机械振动信号特征提取与寿命预测方法研究 [J]. 宇航总体技术，2021：24－32.

[12] 阙养华，江铁强，陈立杰. 基于决策树的多防御目标攻防决策 [J]. 宇航总体技术，2021，5 (5)：35－40.

[13] 王 伟，胡清华，于 宵，等. 多值属性系统的故障诊断出策略最优化方法 [J]. 仪器仪表学报，2008，29 (5)：1073－1078.

[14] DEVASIA S, CHEN D, PADEN B. Nonlinear inversion based output tracking [J]. IEEE Trans on Automatic Control, 1996, 41 (7)：930－942.

[15] DORIGO M. Optimization, Learning and Natural Algorithms [J]. Italy：Politecnico di Milano, 1992, 8 (4)：80－88.

[16] COLORNI A D M. An investigation of some properties of an ant algorithm [J]. Proceedings Of Parallel Problem Solving from Nature (PPSN). France：Elsevier, 1992：509－520.

[17] 景小宁，李全通. 系统维修中的顺序诊断策略 [J]. 电光与控制，2009，16 (1)：87－91.

(上接第 264 页)

[14] 徐东明，卢 斌. 一种改进的 CSA 低功耗阵列乘法器的实现 [J]. 微电子学与计算机，2016，33 (9)：19－23.

[15] 孙偲彦，蒋剑飞，毛志刚. 一种数字信号处理器中的高性能乘加器设计 [J]. 微电子学，2010，40 (1)：32－36.

[16] 唐俊龙，汤孟媛，吴圳羲，等. 32 位 RISC-V 处理器中乘法器的优化设计 [J]. 电子设计工程，2022，30 (6)：61－65.

[17] 王仁平，何明华，魏榕山，等. 32×32 高性能乘法器的全定制设计 [J]. 福州大学学报 (自然科学版)，2012，40 (5)：602－608.

[18] 应琦钢，郑丹丹，何乐年. 基于优化电路的高性能乘法器设计 [J]. 微电子学与计算机，2011，28 (4)：52－56.

[19] WATERMAN A, LEE Y, PATTERSON D A, et al. The RISC-V instruction set manual, volume I：User-level ISA, version 2.2 [R]. Berkeley, CA：University of California, 2020.

[20] BOOTH A D. A signed binary multiplication technique [J]. Q. j. mech. appl. math, 1950 (2)：236－240.

[21] 吴美琪，赵宏亮，刘兴辉，等. 一种基于改进基 4 Booth 算法和 Wallace 树结构的乘法器设计 [J]. 电子设计工程，2019，27 (16)：145－150.

[22] KRITHIVASAN S, KOOB C E, ANDERSON W C. Power-efficient sign extension for booth multiplication methods and systems：US, US7797366 [P]. 2010－9－14.