

用于测试序列优化的 DPSO-WAO * 算法研究

汪芊芊, 林臻, 苏晗, 王海涛, 蓝鲲
(北京宇航系统工程研究所, 北京 100076)

摘要: 针对现有测试序列优化算法所存在的计算效率及优化性能间的矛盾, 在 DPSO-AO * 算法基础上, 采用基于加权 Huffman 编码的启发式评估函数实现对测试成本的评估, 形成了 DPSO-WAO * 算法; 实例证明, 基于加权 Huffman 编码的启发式评估函数更为准确地评估了全局测试成本, 使 DPSO-WAO * 算法优化性能较 DPSO-AO * 算法提升 10% 以上, 趋近传统 AO * 算法, 并通过取消成本回溯有效降低了传统 AO * 算法的计算复杂度, 对于运载火箭等大型系统的测试序列设计、故障诊断等具有重要意义。

关键词: 测试序列优化; 测试代价; 加权 Huffman 编码; 启发式评估函数; DPSO-WAO *

Research on DPSO-WAO * Algorithm Based on Test Sequence Optimization

WANG Qianqian, LIN Zhen, SU Han, WANG Haitao, LAN Kun
(Beijing Institute of Astronautical Systems Engineering, Beijing 100076, China)

Abstract: Aiming at the contradiction between the calculation efficiency and optimization performance of existing test sequence optimization algorithms, on the base of DPSO-AO * algorithm, a heuristic evaluation function based on weighted Huffman coding is used to evaluate testing cost, which proposes a DPSO-WAO * algorithm for testing sequence optimization. The heuristic evaluation function based on weighted Huffman coding is proved by examples that it can more accurately evaluate the global testing cost, and the optimization performance of the DPSO-WAO * algorithm improves over 10% than that of the DPSO-AO * algorithm, approaching the result of traditional AO * algorithm, which effectively reduces the computational complexity of traditional AO * algorithm by canceling the cost back. It is of great significance for the test sequence design and fault diagnosis of large systems such as launch vehicles.

Keywords: test sequence optimization; test cost; weighted Huffman coding; heuristic evaluation function; DPSO-Weight _ AO *

0 引言

由于测试项目对系统资源的占用, 在对系统进行测试的过程中, 所有的测试并不能同时进行, 需对测试进行排序, 而不同的测试执行顺序对于系统的测试效果和测试成本均会产生较大影响^[1-3]。因此, 在执行测试前, 需要对测试序列进行优化设计, 使其尽可能早并精确地隔离故障, 同时降低测试代价 (包括费用、故障隔离时间等综合评价因素), 这正是测试策序列优化设计问题。

针对该问题, 国内外学者提出了优化算法。常用的方法有动态规划 (DP) 算法^[4-6]和启发式算法^[7-13]。动态规划是自底向上构造测试树, 具有指数级的计算量, 无法应用于复杂设备的测试序列优化; 启发式算法是利用启发式评估函数对测试成本进行评估, 以评估结果进行测试点筛选的一类贪婪搜索算法, 目前应用最为广泛的是由 Krishna R. Pattipati 及其团队提出的 AO * (AND/OR *) 算法^[12]。但传统 AO * 算法由于需要通过测试成本回溯实现全局最优, 导致节点扩展量巨大, 当测试个数增加时, 易

导致计算量爆炸。为此, 国内外学者提出了众多改进方法, 主要从两方面考虑: (1) 减少传统 AO * 算法的扩展节点^[14]; (2) 结合现代优化算法降低计算复杂度^[15-20]。以上方法一定程度上降低了传统 AO * 算法的计算复杂度, 但未从全局的角度考虑各个测试项目的代价和故障概率对于测试选择的影响, 导致算法的优化性能下降, 难以达到最佳优化效果。

针对传统 AO * 算法计算量大, 而现有对于 AO * 算法的改进无法同时满足计算效率和优化性能的问题, 本文从启发式评估函数的改进出发, 设计了基于加权 Huffman 编码的启发式评估函数, 对基于离散粒子群的 AO * 算法 (DPSO-AO *) 进行改进, 提出了 DPSO-WAO * (DPSO-Weight _ AO *) 算法。实例证明, 基于加权 Huffman 编码的启发式评估函数较传统 AO * 算法使用的启发式评估函数更为准确地评估了全局测试成本, 在取消了成本回溯的情况下, 算法计算复杂度与 DPSO-AO * 算法相当, 但算法寻优性能较 DPSO-AO * 算法得到了较大提升, 很好地解决了

收稿日期: 2023-06-30; 修回日期: 2023-08-09。

作者简介: 汪芊芊 (1992-), 女, 硕士, 工程师。

引用格式: 汪芊芊, 林臻, 苏晗, 等. 用于测试序列优化的 DPSO-WAO * 算法研究[J]. 计算机测量与控制, 2024, 32(1): 232-236, 274.

传统 AO * 算法存在的计算效率和优化性能的矛盾, 适用于大型复杂对象的测试序列设计。

1 测试序列优化问题分析与建模

对于被测系统来说, 其系统状态和测试信息可以定义为五元组 (S, p, T, c, D), 式中:

$S = \{s_1, s_2, \dots, s_m\}$ 是系统故障状态集合, $s_i (1 \leq i \leq m)$ 表示系统不同的故障状态, 也即故障模式;

$p = [p(s_1), p(s_2), \dots, p(s_m)]^T$ 是系统各故障状态 S 的故障率集合;

$T = \{t_1, t_2, \dots, t_n\}$ 是系统可选的测试集合;

$c = [c_1, c_2, \dots, c_n]^T$ 是各可行测试的测试成本集合, 可以测试时长、人力成本或其它指标评价。

$D_{m \times n}$ 是测试与系统状态的相关性矩阵。其基本形式为:

$$D_{m \times n} = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \dots & \dots & \dots & \dots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{bmatrix} \quad (1)$$

$d_{i,j}$ 表示故障 f_i 与测试 t_j 之间的相关性, 若 $d_{i,j} = 1$, 则表示测试 t_j 可以检测到故障状态 f_i , 若 $d_{i,j} = 0$, 则表示测试 t_j 无法检测到故障状态 f_i 。

假定同一时刻只有一个故障状态 s_i 发生, 且每次只施加一个测试项目, 则测试序列优化就是设计一个方法, 能够通过合理安排测试项目的实施顺序, 来识别系统故障状态集中的故障模式, 且使期望的测试成本最小, 测试成本通过如下公式评估:

$$J = p^T A c = \sum_{i=0}^m \sum_{j=1}^n a_{ij} p(s_i) c_j \quad (2)$$

式 (2) 中 $A_{(m+1) \times n} = [a_{ij}]$ 是一个二值矩阵, 如果在识别系统状态 s_i 的过程中采用了测试 t_j , 则 $a_{ij} = 1$; 否则 $a_{ij} = 0$ 。如何能在满足系统故障隔离率和故障检测率要求的前提下, 使式 (2) 中测试成本最小, 是本文的研究目标。

2 现有 AO * 算法及 DPSO-AO * 算法

AO * 算法通过构建与或图采用 top-down 的搜索方法进行故障隔离, 如图 1 所示。或节点对应故障集节点 X; 与节点对应可用测试节点 t_j , 其各子节点分别代表 t_j 应用于故障状态集节点 X 后根据其测试结果和相关矩阵所得的通过故障子集 X_{jp} 和失败故障子集 X_{jf} 。

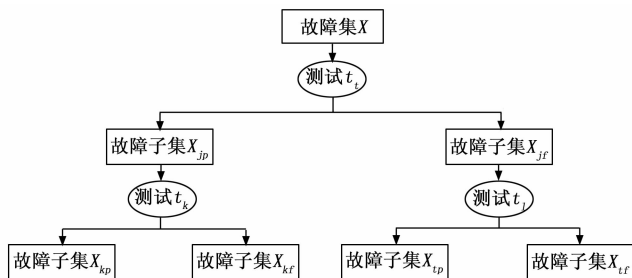


图 1 AO * 算法应用于测试序列优化示例

AO * 算法是两个运算阶段的反复:

1) 自上而下进行节点扩展。从初始模拟故障集开始, 评估所有可选测试用于隔离模糊故障集的测试代价, 并选取代价最小的测试进行与或树扩展, 直至所有故障集节点均为单节点, 则解树生成, 实现故障隔离的目的。

2) 自下而上进行成本回溯。在与或树扩展过程中, 单纯根据局部极值进行节点扩展, 可能陷入局部最优, 因此, 需要在每一次节点扩展后自下而上进行成本回溯, 计算初始模拟故障集的测试成本, 并与该次扩展前的成本评估值比较, 若相等, 则继续扩展, 否则, 选取启发式函数值次小的测试进行该节点的扩展, 直至回溯成本与扩展前成本相同 (若各可用测试所达到的成本回溯值均大于扩展前的评估值, 则选取回溯成本最小的测试进行与或树扩展)。

AO * 算法利用 Huffman 编码构造启发式评估函数。假设测试费用满足: $0 \leq c_1 \leq \dots \leq c_n$, 构造评估隔离故障集 X 的最小平均测试代价启发式函数如式 (3) 所示:

$$HEF(X) = \sum_{j=1}^n c_j + [\omega^*(x) - \omega'(x)] \times c_{\omega'(x)} \quad (3)$$

其中: $\omega'(x)$ 表示 $\omega^*(x)$ 的整数部分。 $\omega^*(x) = [p(x)]^{-1} \sum_{s_i} \omega(s_i) p(s_i)$, $\omega(s_i)$ 表示 X 中状态 s_i 的 Huffman 编码。

若故障集 X 被测试 t_j 区分为两个故障子集 X_{jp} 和 X_{jf} ($X_{jp} \cup X_{jf} = X$), 其中 X_{jp} 表示测试 t_j 无法检测到的故障子集, X_{jf} 表示测试 t_j 可以检测到的故障子集。利用测试点 t_j 隔离故障集 X 的最小测试代价近似估计为:

$$h_x = c_j + p(x_{jp}) \times h(x_{jp}) + p(x_{jf}) \times h(x_{jf}) \quad (4)$$

其中: $h(x_{jp})$ 和 $h(x_{jf})$ 分别表示估计隔离故障集 X_{jp} 和 X_{jf} 的最小测试代价, 利用 $HEF(X)$ 估计; $p(x_{jp})$ 和 $p(x_{jf})$ 分别表示测试 t_j 通过和失败的相对概率。传统 AO * 算法利用式 (4) 中 h_x 的值评价测试点 t_j , 选取 h_x 最小的 t_j 向下扩展。并在每次向下扩展之后将测试成本向上回溯, 以确保当前所扩展测试的全局最优性。

传统 AO * 算法通过不断回溯以寻得最优测试序列, 对于运载火箭电气系统、电气设备这样规模庞大的优化对象, 由于涉及的测试项目、故障模式均较多, 因此所需运算时间急剧增加, 很容易出现计算爆炸问题。为解决传统 AO * 算法回溯次数多, 计算量大的问题, 文献 [18] 将离散粒子群算法 (DPSO) 与 AO * 算法相结合, 对 AO * 算法的可扩展节点进行筛选, 并取消了成本回溯, 大大减少了 AO * 算法的计算复杂度。

DPSO-AO * 算法的基本思想是^[18,20]: 利用 DPSO 算法优选 AO * 算法中每个待扩展节点的测试集, 进而减少待扩展测试的个数, 并取消成本回溯, 降低算法计算复杂度。DPSO-AO * 算法采用式 (5) 所示的适应度函数对备选测试集 T 进行优化:

$$F(T_s) = \eta_{FD} \eta_{FI} \left(\frac{\sum_{t_i \in T_s} h(t_i) / \sum_{t_i \in T_s} c_i}{N(\sum_{t_j \in T} h(t_j) / \sum_{t_j \in T} c_j)} \right)^2 \quad (5)$$

其中： T_s 代表所选的最优测试集， N 代表测试集 T_s 中测试项目的个数， η_{FD} 、 η_{FI} 分别代表 T_s 的故障检测率和故障隔离率； c_i 、 c_j 分别代表测试 t_i 、 t_j 的测试代价； $h(t_i)$ 代表测试集 T 的单个测试 t_i 的优劣程度，如式 (6) 所示：

$$h(t_i) = a \times \frac{P(t_i)}{S(t_i)} \quad (6)$$

式 (6) 中， $P(t_i)$ 代表测试 t_i 能检测到的故障可能发生的总概率； $S(t_i)$ 代表 t_i 所能检测到的故障的最小可测度， a 为其系数，需根据实际优化结果进行调整。

3 基于改进评估函数的 DPSO-WAO * 算法

3.1 基于加权 Huffman 编码的启发式评估函数

传统 AO * 算法依据评估函数选择最优测试，实现故障树的向下扩展，并结合向上回溯来保证测试选择的正确性，其优化性能主要取决于以下两点。

1) 启发式评估函数对于故障集的测试代价描述的准确性：

启发式评估函数越接近真实的最小测试代价，则对于测试的选择越准确，越接近全局最优。

2) 回溯步数：

回溯的目的是检验当前所选测试的全局最优性，向上回溯步数越多，则越接近全局最优，若每一次测试选择都向顶端节点回溯，则所生成的测试序列一定为全局最优。

传统 AO * 算法利用向顶端节点回溯来确保算法的全局优化性能，但回溯步数过多会导致扩展节点数目增多，因此计算量巨大，不适用于大规模系统的测试序列优化。DPSO-AO * 算法结合离散粒子群算法对传统 AO * 算法进行改进，通过 DPSO 算法对扩展测试集进行优选，并取消成本回溯来减少节点扩展数目，降低计算复杂度。DPSO-AO * 算法所用的启发式函数 (3) 主要通过故障集中各个故障状态的平均 Huffman 编码来确定所需测试数量，并按照测试成本由低到高的原则选取测试项目用于计算总测试成本。

在理论编码中，Huffman 树 (最优二叉树) 具有带权路径长度最小 (即成本最小) 的特点，因此，可用于近似估计测试集的最小测试成本。但该方法忽略了每个故障状态的 Huffman 编码信息，而是求取各故障状态的平均 Huffman 编码用于评估最小测试代价。事实上，Huffman 编码越短的故障，对应故障率越高，越应该被提前隔离，而仅通过平均 Huffman 编码，无法体现该原则。因此，本文在 DPSO-AO * 算法的基础上，从启发式评估函数的改进这一角度出发，构建了基于加权 Huffman 编码的启发式函数，对 DPSO-AO * 算法进行改进。加权 Huffman 编码的启发式函数基本思想如下：

隔离系统各状态的测试总代价是通过将隔离各个状态的测试代价相叠加而得的，而各个状态的测试代价为隔离该状态所需的测试代价与该故障状态的故障率的乘积。因此，在对某个模糊状态集的测试代价进行评估时，分别考虑模糊集中的各个状态，利用故障率作为故障模式 Huff-

man 编码的权重，提出如下基于加权 Huffman 编码的启发式函数：

$$HEF'(X) = [p(x)]^{-1} \sum_{s_i \in X} (\sum_{j=1}^{\omega(s_i)} c_j \times p(s_i)) \quad (7)$$

其中：测试费用满足 $0 \leq c_1 \leq \dots \leq c_n$ ， $\omega(s_i)$ 表示故障模糊集 X 中 s_i 的 Huffman 编码长度。

式 (7) 所示的基于加权 Huffman 编码的启发式函数表述的诊断树针对每个故障模式分别计算所需的测试数量。 $\sum_{j=1}^{\omega(s_i)} c_j \times p(s_i)$ 表征隔离故障模式 s_i 所需的最小测试成本。基于加权 Huffman 编码的启发式函数同时满足以下测试序列优化的原则：

- 1) 优先隔离故障率高的故障；
- 2) 优先选择测试成本低的测试。

由测试成本定义式 (2) 可知，该值即为测试成本的下限值。

基于加权 Huffman 编码的启发式函数与传统的启发式函数的差别在于：传统 AO * 算法利用模糊故障集的平均 Huffman 编码来表征隔离故障集的平均最少测试步骤，而丢失了故障集中的隔离各个故障所需的最少测试步骤这一信息，未体现优先隔离故障率高的故障的这一原则。而基于加权 Huffman 编码的启发式函数构建了满足最优测试序列优化原则的诊断树，并准确计算了该最优诊断树的测试代价。因此，利用基于加权 Huffman 编码的启发式函数评估测试代价更为准确。

3.2 DPSO-WAO * 算法实施步骤

基于 DPSO-WAO * 算法的测试序列优化方法与传统 AO * 算法基本相同，但由于启发式函数的准确性更高，因此取消了向上回溯这一步骤，流程如图 2 所示。

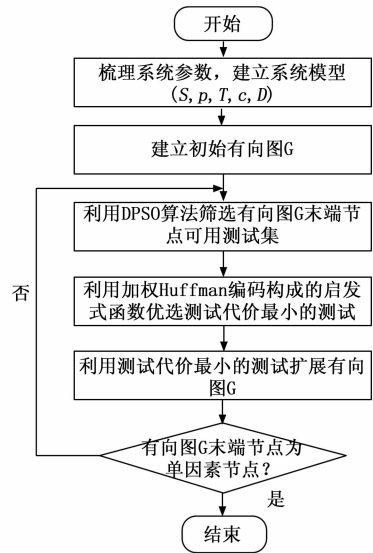


图 2 基于 DPSO-WAO * 算法的测试序列优化流程

具体流程如下：

- 1) 建立一个仅由根节点 (系统故障状态集) 构成的有

向图。

2) 对于模糊故障集 (初始值为系统故障状态集), 利用 DPSO 算法对当前可用测试集进行优选; 利用评估函数评估采用各优选出的测试隔离当前模糊故障集的测试代价, 并选出测试代价最低的测试扩展有向图, 直到中仅含有一个元素。具体步骤如下:

① 利用 DPSO 算法对当前可用测试集进行筛选, 获得优化的测试子集;

② 在测试子集中, 每个测试将模糊故障集 (模糊故障集的初始值) 分为左右两个子集和, 由式 (7) 估计两个子集的最优隔离代价, 由式 (4) 计算利用测试隔离模糊故障集的最小代价;

③ 找出测试子集中隔离模糊故障集所用代价最小的测试, 利用该测试扩展有向图, 获得左右两个子集;

④ 分别以故障子集作为模糊故障集, 重复 2) 进行有向图的扩展, 直至为单元集合, 此时, 系统故障状态集中各个状态均能被区分。

4 实例校验及算法对比

4.1 实例校验

为验证 DPSO-WAO* 算法的有效性, 在 matlab 平台上实现了 DPSO-WAO* 算法, 并利用该算法对文献 [18] 中的超外差接收器系统进行测试序列优化工作, 与文献 [18] 提出的 DPSO-AO* 算法进行比对。

文献 [18] 所给出的超外差接收器系统包括 22 个不同的系统状态, 各状态发生概率不等; 包括 36 种测试, 各测试的成本相同, 均设为 1。

然而, 在实际测试条件下, 各个测试的成本并不完全相同, 由测试难易程度、测试时间、测试人力成本、资金成本等影响因素决定, 对最终的系统测试总成本产生较大的影响。因此, 本文对文献 [18] 中给出的超外差接收器系统的 36 个测试项目赋予测试成本属性, 形成超外差接收器测试成本集合 c_{jsq} , 如式 (8) 所示, 测试成本从 2 到 67 不等, 数值越高代表测试成本越高。

$$c_{jsq} = [10, 25, 2, 7, 10, 10, 30, 29, 30, 50, 50, 20, 4, 5, 67, 34, 30, 50, 50, 20, 4, 20, 4, 5, 67, 20, 4, 5, 67, 29, 30, 50, 50, 45, 5, 67]^T \quad (8)$$

假设该系统同一时刻只有一个故障发生, 且同一时刻只进行一种测试。对于测试成本相同及测试成本不同的情况, 在 matlab 平台上分别利用传统 AO* 算法、DPSO-AO* 算法、DPSO-WAO* 算法对式 (8) 描述的系统进行测试序列优化。其中, DPSO 的粒子数目取 15, 迭代次数取 100 时, 优化性能较好。通过 3 种不同优化算法形成的优化结果如表 1 及图 3、图 4 所示, 表 1 中的数据为对应算法连续运行 10 次所得结果的平均值或标准差。

分别比较了测试成本相同 (均为 1) 和测试成本不同 (各项测试成本设置参见式 (8)) 两种工况下, 3 种算法寻得的最小平均测试代价、测试代价标准差、平均允许运行时间及平均扩展节点数目。其中:

1) 平均最小测试代价。代表算法优化性能, 该数值越小, 则代表算法优化得出的测试代价越小, 优化性能越好。

2) 测试代价标准差。代表算法寻优性能的稳定性, 该数值越小, 代表各次优化结果一致性好, 算法运行稳定可靠。

3) 平均运行时间。代表算法计算效率, 该数值越小, 代表算法运行时间越少, 计算效率越高。

4) 平均扩展节点数目。代表算法计算复杂度, 该数值越小, 代表算法遍历次数越少, 算法复杂度越低。

从表 1 结果可以看出, DPSO-WAO* 算法可有效寻得最小测试代价, 算法性能稳定, 运算效率较高。

表 1 各算法性能参数比较

| 优化场景 | 算法名称 | 平均最小测试代价 | 测试代价标准差 | 平均运行时间/s (matlab 环境下) | 平均扩展节点数目 |
|------------------------------|--------------|----------|---------|-----------------------|----------|
| 测试成本相同时各算法性能比较 (各测试成本均为 1) | 传统 AO* 算法 | 3.35 | 0 | 68.4 | 976 |
| | DPSO-AO* 算法 | 3.77 | 0.013 9 | 13.1 | 246 |
| | DPSO-WAO* 算法 | 3.37 | 0.010 3 | 12.0 | 228 |
| 测试成本不同时各算法性能比较 (各测试成本参见式(8)) | 传统 AO* 算法 | 42.02 | 0 | 87.2 | 1 076 |
| | DPSO-AO* 算法 | 48.05 | 1.572 9 | 12.3 | 145 |
| | DPSO-WAO* 算法 | 43.66 | 0.881 1 | 12.6 | 142 |

注: 算法各性能参数均以连续运行 10 次所得结果取平均值。

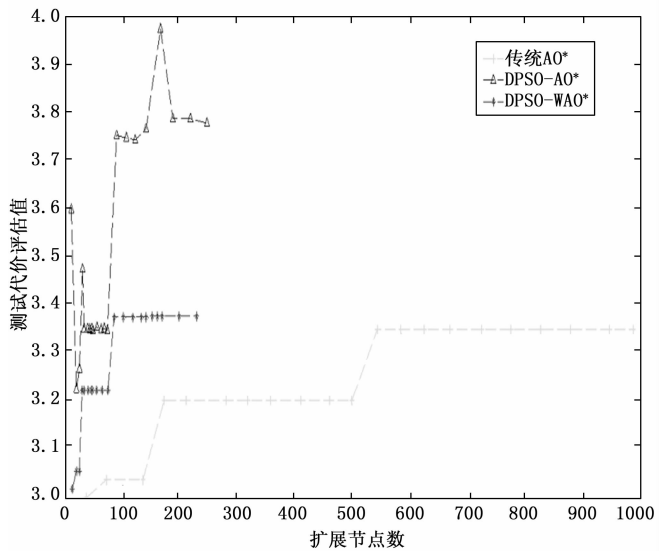


图 3 测试序列优化算法性能比对 (测试成本相同)

4.2 算法比对

上文通过超外差接收器系统对 DPSO-WAO* 算法有效性进行了实例验证, 并与传统 AO* 算法、文献 [18] 提出的 DPSO-AO* 算法进行了性能比对, 如表 1 及图 3、图 4 所示, 得到结论如下。

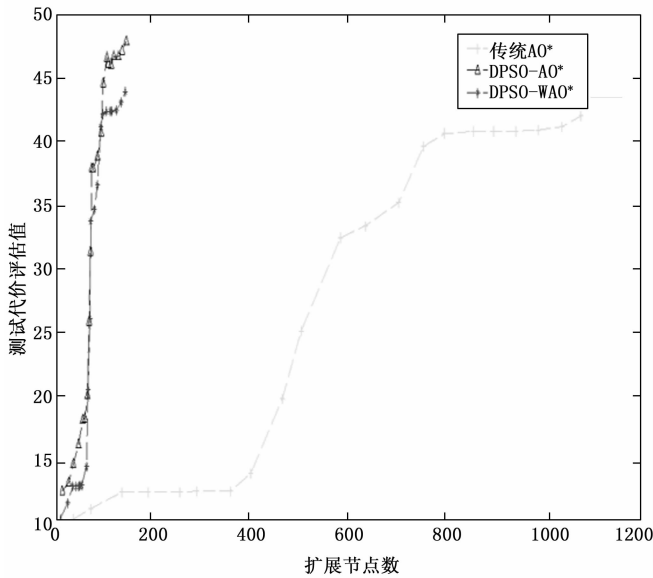


图 4 测试序列优化算法性能比对 (测试成本不同)

1) 寻优性能:

DPSO-WAO* 算法的优化性能稍差于传统 AO* 算法, 但二者优化生成的测试序列的测试代价相差不超过 3.9%, 主要是由于传统 AO* 算法通过不断向上回溯找到全局最优解, 而 DPSO-WAO* 算法无回溯, 得到的是近似全局最优解, 此结果与预期一致。

DPSO-WAO* 算法与 DPSO-AO* 算法相比, 生成的最小测试代价分别降低 11.9% (测试代价相同) 和 10.1% (测试成本不同), 寻优性能得到较大提升。由此证明, DPSO-WAO* 算法同 DPSO-AO* 算法一样, 均取消了传统 AO* 算法的回溯, 但由于基于加权 Huffman 编码的启发式函数能更好的评估模糊故障集的隔离代价, 使当前测试节点的扩展路径更接近全局最优, 因此, DPSO-WAO* 算法优化性能较 DPSO-AO* 算法可提升 10% 以上。

2) 计算效率和复杂度:

DPSO-WAO* 算法的平均运行时间与 DPSO-AO* 算法的平均运行时间相当, 且均远小于传统 AO* 算法, 仅为传统算法所需时间的约 15%, 其扩展节点数目也约为传统 AO* 算法的 15%。可以证明, DPSO-WAO* 算法在进行测试节点扩展前利用 DPSO 对当前可用测试集进行优选, 避免陷入局部最优, 并大大减少了扩展节点的数目, 提高了算法的计算效率, 且越复杂的系统, 采用 DPSO-WAO* 算法获得的计算效率提升效果也越明显。

5 测试序列优化方法在运载火箭电子设备中的应用模式思考

运载火箭电气系统组成复杂, 故障模式在数量和种类上呈现多样性, 需采取较多测试才能实现故障检测及隔离需求, 如果不对测试进行统筹规划, 将会耗费大量的时间和成本代价。现阶段运载火箭电子设备的测试包括单机测试、系统综合测试、全箭电气测试等多个环节, 各测试环节主要关注测试覆盖性, 及测试项目的设置需尽可能覆盖

产品的功能、性能、冗余设计和对外接口等, 测试项目的设置和排序主要依赖测试人员经验, 未考虑测试对故障的隔离定位能力、测试成本等影响因素。

随着新型火箭的研制, 传统火箭测试设计思路已很难适应新型火箭的测试需求, 尤其是随着智慧火箭、可回收火箭等新概念的提出, 传统依靠人工经验通过断插头、串转接盒等测试、排故模式已经无法适应火箭的测试排故需求, 新型火箭的测试项目和测试序列的设计逐步向如下趋势发展:

1) 从地面 ATE 测试、人工测试模式转向箭上 BIT 测试模式;

2) 测试项目的选择和排序从依据人员经验和定性原则转向依据定量优化方法。

对于运载火箭来说, 测试序列优化设计可实现在常规测试或排故时, 对测试项目进行优选并形成最优测试序列, 达到快速、自动地识别故障并将故障隔离定位到设定的故障隔离层次, 如功能电路层、板卡层、单机层的目的。新研火箭电气系统设计可以在地面测试设备中嵌入测试序列优化设计功能, 实现从测试序列自动生成、到故障诊断模型生成、到测试程序集生成、到测试设备测试操作、最后到故障诊断报告生成的自动测试排故流程, 辅助测试人员开展自动故障排查, 减少当前凭经验排故带来的故障定位困难及大量的人力、时间成本, 提高排故效率和自动化程度, 适应新型火箭的快速测试发射和排故需求。

6 结束语

通过对算法各性能参数的分析可以发现, 基于加权 Huffman 编码的启发式评估函数能够更准确地评估隔离模糊故障集的测试代价, 使当前测试节点的扩展路径更接近全局最优, 对于测试成本不同时的测试序列优化具有更为明显的效果。因此, DPSO-WAO* 算法虽然取消了传统 AO* 算法的回溯, 但仍然取得了很好的优化效果, 并且大大降低了计算复杂度, 同时较 DPSO-AO* 算法实现了寻优性能的提升, 可应用于运载火箭、武器装备等大型复杂系统的测试序列设计、故障诊断及定位中, 具有重要意义。

参考文献:

- [1] 彭越, 牟宇, 宋敬群. 中国下一代运载火箭电气系统技术发展研究 [J]. 宇航总体技术, 2020, 4 (2): 13-24.
- [2] 景小宁, 李全通, 陈云翔, 等. 基于信息熵的最少测试费用故障诊断策略 [J]. 计算机应用, 2005, 25 (2): 417-419.
- [3] 于劲松, 张帆, 万九卿. 基于故障相关矩阵的最优测试序列生成方法 [J]. 计算机测量与控制, 2009, 16 (4): 1077-1083.
- [4] 苏永定. 机电产品测试性辅助分析与决策相关技术研究 [D]. 长沙: 国防科学技术大学, 2004.
- [5] 罗建军, 王嘉文, 王明明, 等. 机器人在轨构建空间桁架结构的装配序列规划方法 [J]. 宇航学报, 2021, 42 (4): 437-449.

(下转第 274 页)