

基于 DPSO 的改进 AO* 算法在大型复杂电子系统最优序贯测试中的应用

蒋荣华 王厚军 龙 兵

(电子科技大学自动化学院 成都 610054)

摘 要 针对大型复杂电子系统最优序贯测试问题,提出一种基于离散粒子群算法(DPSO)和改进 AO* 算法相结合的方法. DPSO 优化 AO* 算法中每个要扩展节点的测试集从而减少测试个数;改进 AO* 算法通过规定扩展节点估价值的范围,减少其回溯次数. 实例验证表明,该算法不仅有效地降低了计算复杂度,大大减少测试代价,缩短测试时间,而且避免了原有 AO* 算法当备选的测试集太大时容易出现“计算爆炸”的缺点.

关键词 离散粒子群算法;AO* 算法;序贯测试;哈夫曼编码;可测性设计

中图法分类号 TP301

Applying Improved AO* Based on DPSO Algorithm in the Optimal Test-Sequencing Problem of Large-scale Complicated Electronic System

JIANG Rong-Hua WANG Hou-Jun LONG Bin

(School of Automation, University of Electronics Science and Technology of China, Chengdu 610054)

Abstract An algorithm of improved AO* based on discrete binary particle swarm optimization (DPSO) is proposed, which can solve the Optimal Test-sequencing problem in large-scale complicated electron system. DPSO optimizes the test sets which can isolate the expanded node in AO* algorithm to decrease the number of node; The improved AO* limits the test cost range of node and lessens the traces. The result of real operation show that this algorithm not only reduces the computational complexity, cuts down the test cost, shorten the test time; but also avoids the "computational explosion" when the test set is too large.

Keywords discrete binary particle swarm optimization; AO* Algorithm; test sequence; Huffman coding; design for testability

1 引 言

最优序贯测试问题是可测性设计与分析方法中的一个关键问题,已证明它是一个 NP 完全问题^[1],目前已有的求解该问题的方法主要有两类:动态规划和启发式搜索. 动态规划是自底向上构造测试树,算法具有指数级存储及计算复杂度 $O(3^n)$ (n 为备

选的测试数目),因此不适于测试数 $n > 12$ 的情况^[1];Raghavan^[2]和 Shakeri^[3]等人提出采用基于熵启发式搜索方法,通过构建与或图采用 top-down 的搜索方法进行故障隔离(AO* 算法),通过 AO* 算法每一步扩展节点的熵选择当前最优测试点,是一种局部的逐步优化的“贪心算法”,但该算法需要多次回溯才能得到最优的测试序列,又由于该算法是局部优化算法,容易陷入局部最优,可能得不到全局

最优解;2003年,Tu^[4]提出采用拉格朗日松弛与次梯度来解决最优测试问题,计算效率得到进一步提高,但当系统规模较大时,其计算效率仍然不高;2004年,文献[5]提出根据故障概率、测试费用以及测试与故障间的约束关系(故障字典),应用遗传算法生成最优的诊断测试策略,可有效降低测试代价,适用于大规模系统,但由于智能算法本身依然需要多次迭代才能获得最优的测试顺序,所需时间较长;2007年,Kennedy^[6]提出一种 bottom-up 的测试序列生成方法,虽然克服了原有 AO* 算法容易陷入局部最优的缺点,但是算法需要每次进行迭代才能找到全局最优解,其运算时间也随着问题规模的扩大而急剧增大,而且容易出现‘组合爆炸’.目前随着系统集成度的提高、规模的扩大,迫切需要一种既快速又能找到全局最优的序贯测试算法.

为此,本文针对 AO* 算法产生回溯次数多,不容易找到全局最优解的缺点,根据粒子群算法(PSO)收敛速度快、全局优化性能好的特点,提出了使用离散粒子群算法(DPSO)与改进的 AO* 算法(简称 DPSO-AO*)相结合的方法来处理大型复杂系统最优序贯测试问题. DPSO 是一种新型智能算法,目前,将其用于测试问题的求解还很少,本文算法不仅有效地将智能算法 DPSO 运用于测试选择问题,而且克服了原有 AO* 算法当备选测试集太大时,容易出现“计算爆炸”的缺点,从而有效地提高了计算效率,对于大型系统的可测性设计与分析、故障诊断及粒子群算法的应用等具有很重要的意义.

2 问题描述

最优序贯测试问题(也称之为测试规划问题)可以定义为五元组 (S, p, T, c, D) , 式中 $S = \{s_0, s_1, s_2, \dots, s_m\}$ 是与系统状态相关的有限集, 其中 s_0 表示“无故障”状态, $s_i (1 \leq i \leq m)$ 表示系统不同的故障状态; $p = [p(s_0), p(s_1), \dots, p(s_m)]^T$ 是系统状态 S 的先验概率矢量; $T = \{t_1, t_2, \dots, t_n\}$ 是 n 个可行测试; $C = [c_1, c_2, \dots, c_n]^T$ 是以时间、人力要求或其它经济指标衡量的测试成本矢量. D 是一个 $(m+1) \times n$ 维的二值矩阵, 它代表测试与系统状态的关联关系, 如果 D 的 i 行元素 d_{ij} 是 1, 则测试 $t_j (1 \leq j \leq n)$ 可以检测到故障源 s_i ; 若 d_{ij} 是 0, 则表示故障状态 s_i 发生而测试 t_j 不报警, 显然 $d_{0j} = 0 (1 \leq j \leq n)$. 假定只有一个系统状态 $s_i (1 \leq i \leq m)$ 发生, 而且给定二值依赖矩阵 $D = [d_{ij}]$. 序贯测试就是设计一个测试算法能够采用测试集 T 的测试, 确定地识别出 S 中的某系统状

态, 且使期望的测试成本最小, 其值为

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

式中 $A = (a_{ij})$ 是 $(m+1)$ 乘 n 的二值矩阵, 如果在识别系统状态过程中采用了测试 t_j , 则 $a_{ij} = 1$; 否则 $a_{ij} = 0$. 如何使选出的序贯测试集, 在满足故障隔离率和故障检测率为 1 的情况下, 使式(1)的值最小, 测试时间最短, 是本文的研究目标.

3 算法介绍

3.1 DPSO 算法

1995年,Kennedy和Eberhart^[7]提出基于群体中的个体(如粒子)行为及数学抽象的 PSO 算法, 算法采用速度-位置模型, 即 PSO 算法在允许范围内初始化为一群随机粒子(潜在解), 每个粒子都有一个速度决定它们的飞行方向和距离, 在每一次迭代中通过跟踪两个极值来更新自己: 粒子本身迄今为止所找到的个体极值 $Pbest_{id}$ 和整个种群迄今为止所找到的全局极值 $Gbest_{id}$. 所有粒子的优劣由被优化函数所决定的适应度来衡量^[7].

可解决离散空间问题的 DPSO 是 Kennedy 和 Eberhart 在 1997 年提出的^[8], 粒子的速度和位置公式如式(2)和式(3):

$$v_{id}^{t+1} = \omega v_{id}^t + c_1 r_1^t (P_{bestid}^t - x_{id}^t) + c_2 r_2^t (G_{bestid}^t - x_{id}^t) \quad (2)$$

$$\begin{cases} x_{id}^t = \begin{cases} 0, & rand \geq \text{sig}(v_{id}^{t+1}) \\ 1, & \text{其它} \end{cases} \\ \text{sig}(v_{id}^{t+1}) = 1 / (1 + \exp(-v_{id}^{t+1})) \end{cases} \quad (3)$$

式(2)中, v_{id}^t 为粒子 i 在迭代第 t 次的速度; ω 是为避免 DPSO 陷入局部最优而引入的惯性权重因子^[9], 本文取 $\omega = 1.2$ 取得了较好效果, $c_j (j=1, 2)$ 为加速常数; r_1^t, r_2^t 是 0 到 1 之间的随机数; x_{id}^t 为个体 i 在迭代第 t 次时的当前位置; $Pbest_{id}$ 为第 i 个粒子的个体极值; $Gbest_{id}$ 为全局极值. 式(3)中: $rand$ 为 0~1 的随机数, $\text{sig}()$ 是一个根据粒子速度控制粒子位置为 1 或 0 的函数.

3.2 AO* 算法

AO* 算法属于与/或树启发式搜索算法, 在搜索的过程中能充分利用与问题有关的特征信息, 估计出待选测试的重要性, 问题特征由启发函数来反映, 而测试的重要性由估价函数表现.

与/或树中的初始节点对应待诊断的系统; 叶子节点对应待隔离的各系统状态; 或节点对应故障集节点 S ; 与节点对应可用测试节点 t_j , 测试代价为 c_j ; 其各子节点分别代表 t_j 应用于节点 S 后根据其

测试结果和相关矩阵所得的通过故障子集 S_{jp} 和失败故障子集 S_{jf} , 则 S 估价函数由式(4)给出:

$$h(x) = c_j + p(S_{jp})h(S_{jp}) + p(S_{jf})h(S_{jf}) \quad (4)$$

其中, $p(S_{jp}), p(S_{jf})$ 分别由式(5)、(6)给出:

$$p(S_{jp}) = \left[\sum_{s_j \in S} (1 - d_{ij}) p(s_i) \right] \cdot [\hat{p}(S)] \quad (5)$$

$$p(S_{jf}) = 1 - p(S_{jp}), \quad \hat{p}(S) = \sum_{s_j \in S} p(s_i) \quad (6)$$

其中 $h(S_{jp}), h(S_{jf})$ 表示故障子集 S_{jp} 和失败故障子集 S_{jf} 的启发函数。

根据信息编码理论与序贯测试问题的相似性, 可采用 Huffman 编码和平均信息熵构造启发函数, 由文献[1]可知, 基于 Huffman 编码取得了较好效果, 本文采用式(7)作为启发函数:

$$HEF = \sum_{j=1}^{w'(x)} c_j + [\omega^*(S) - w'(S)] c_{|\omega^*(S)|-1} \quad (7)$$

其中 $\omega^*(S)$ 为 S 的 Huffman 平均码长, 由式(8)给出:

$$\omega^*(S) = [\hat{p}(S)]^{-1} \sum_{s_i \in S} \omega_i^* p(s_i) \quad (8)$$

ω_i^* 为状态 s_i 对应于 S 的 Huffman 编码的长度。

4 基于 DPSO-AO* 算法的序贯测试实现方法

DPSO-AO* 序贯测试的基本思想是: 利用 DPSO 优化 AO* 算法中每个要扩展节点的测试集, 进而减少测试个数, 通过 AO* 算法规定扩展节点估价值的范围, 减少其回溯次数, 最终实现优化的序贯测试。

4.1 DPSO 与测试集优化问题的结合方法

如何将 DPSO 与测试选择问题相结合, 这是本文的一个难点。由前面可知, 粒子群算法与实际问题的结合点是粒子适应度函数, 而粒子就是问题的潜在解, 所以本文根据测试选择的实际问题做了如下定义:

(1) 粒子重定义

所谓的粒子, 就是指所要求解的目标的位置, 由于我们所求的测试集(也就是粒子所要找的解)将组成一个多维的矩阵, 所以我们的粒子也必须是一个多维矩阵, 因此, 在本文把粒子定义为和依赖矩阵 A 同维数的二进制矩阵 X , x_{ij} 第 i 个故障是否选用测试 t_i 进行测试, 如果选用则为 1, 否则为 0。

(2) 粒子速度重定义

粒子速度将决定粒子下一次所在的位置, 所以本文中粒子速度定义为和粒子同维数的随机矩阵, 由于在 DPSO 算法中, 粒子速度只是决定粒子位置变更的概率, 所以此矩阵不必是二进制矩阵。

(3) 粒子适应度函数定义

粒子适应度值将决定粒子是否能被选中, 对于测试选择, 如何把故障检测率、故障隔离率及测试费用融入粒子适应度函数, 这是粒子适应度函数要解决的主要问题。

一个故障的可测度越小, 其被测试到的概率就越小。设故障 s_i 只能被 t_i 和 t_j 检测到, 其可测度为 2, 若排除 t_i, t_j 成为必选测试, 否则该故障就不可测了, 这就加大了对下一步测试的约束。因此, 选择覆盖度小的测试有利于检测和隔离可测度小的故障及下一步的测试选择, 从而保证测试集的故障检测率和隔离率^[7]。

对于测试集 T 中的单个测试 t_i 的优劣程度, 可用如下启发函数来进行衡量:

$$h(t_i) = a \times \frac{P(t_i)N(t_i)}{S(t_i)} \quad (9)$$

式(9)中, $N(t_i)$ 为测试 t_i 能检测到故障集中的故障数; $P(t_i)$ 为测试 t_i 能检测到的故障可能发生的平均概率; 如果 $P(t_i)$ 越大, 也就是 t_i 所能检测到的故障发生的概率较大, 则证明测试 t_i 越有价值; $S(t_i)$ 表示 t_i 所能检测到的故障集中所有故障的最小可测度。某故障的可测度表示, 该故障能被测试集中测试检测到的个数; 式中 a 为其系数, 如果希望最优测试集的故障检测率高, 则可使 a 在 0~1 范围内取较小值, 如果希望最优测试集包含的测试个数少, 且能检测大概率故障, 则可使 a 在大于 1 的范围内取较大值, 本文取 $a=1.2$ 得到了较好效果。 $h(t_i)$ 与 $N(t_i)$ 成正比, 与 $S(t_i)$ 成反比。因此启发函数值越大, 测试优先级就越高。

但是对于所选择的整个测试集的优劣程度, 不仅要考虑单个测试的故障隔离率、检测率, 还需要考虑整个测试集的测试代价及测试集个数等因素, 为此本文采用式(10)的适应度函数:

$$F(T_s) = \eta_{FD} \eta_{FI} \left[\frac{\left(\sum_{t_i \in T_s} h(t_i) / \sum_{t_j \in T_s} c_j \right)}{N \left(\sum_{t_j \in T} h(t_j) / \sum_{t_j \in T} c_j \right)} \right]^2 \quad (10)$$

式(10)中 T_s 表示所选的最优测试集, T 表示备选测试集; N 表示 T_s 的个数, 用于计算 T_s 的测试代价与启发函数值的平均值, 以防止 T_s 的测试个数太多; η_{FD}, η_{FI} 分别表示 T_s 的故障检测率和故障隔离率; c_i, c_j 分别表示测试 t_i, t_j 的测试代价; $h(t_i), h(t_j)$ 分别表示测试 t_i, t_j 的启发函数值。从各参数的含义可知: 粒子适应度函数 $F(T_s)$ 值越大越好。

4.2 AO* 算法改进

文献[1]中的 AO* 算法属于局部寻优算法, 需要扩展较多的节点才能获得最优解, 且该算法在节点扩展过程中不断修正根节点的测试代价, 产生回

溯,降低了算法效率.而本文引入的 DPSO 算法具有全局寻优能力,对 AO* 每个要扩展节点通过 DPSO 在所有测点中进行优选,保证了 AO* 每个扩展节点的全局最优性,因此,本文提出采用限定每一个扩展节点测试估价值范围的方法(AO_δ*),当要扩展的节点估价值超出设定范围,则不再继续扩展,最终达到减少扩展节点数目和全局寻优目的.

4.3 DPSO-AO_δ* 最优序贯测试步骤

使用 DPSO-AO_δ* 的方法求解最优序贯测试问题的具体步骤如下.

1. 应用离散粒子群算法首先对根节点(系统状态集 S)所需的测试集进行优化,得到子优化集 T_r ;
2. 应用 AO* 算法,对根接点进行扩展,通过 AO* 的启发式函数在 T_r 中选出估价函数值最小的节点作为下一个要扩展的对象 S_1 ;具体步骤如下:
 - 2.1. 对测试集 T_r ,每个测试 t_i 将测试集分为左右两个子集 S_{j_p} 和 S_{j_f} ,由式(7)计算两个子集的概率 $P(S_{j_p})$ 和 $P(S_{j_f})$,由式(8)计算两个子集的估价 $h(S_{j_p})$ 和 $h(S_{j_f})$.
 - 2.2. 找到测试集 T_r 中能使式(5)最小的测试 t_i ,得到 S 的当前估价值 $h(S)$.
 - 2.3. 选择步 2 中测试 t_i 所得到的两个子集中启发函数值较大的节点(其估价为 $h(S_i)$)作为新的扩展对象 S_1 .

3. 应用 DPSO 在 T_r (去掉 t_i) 中求解要识别 S_1 状态的子优化集 T_r^1 .
4. 通过 AO* 继续对 S_1 扩展,重复步 2.1.
5. 通过步 4 得到 T_r^1 的每个测试对 S_1 的估价值与 S_1 原估价 $h(S_i)$ 作差值比较,其差值为 δ .
6. 找到最小 δ ,如果 δ 没超出规定值 ϵ ,则找到 δ 最小的测试 t_k 所得到的两个子集中启发函数值较大的节点(其估价为 $h(S_i^1)$)作为新的扩展对象 S_2 ,原 S_1 的估价变为 $h(S_{ik})$. 否则,不再扩展该节点,重复步 2,找新的节点进行扩展.
7. 重复步 3~6,直到 S 所有状态均能被区分.

5 DPSO-AO_δ* 算法验证

为验证其有效性,在 Pentium IV、内存 256MB 和 windows XP 系统环境下,实现了该算法,并对文献[1]中的超出差接受器系统进行序贯测试,该系统有 22 个先验故障和 36 个预备测试,测试代价都为 1,其关联矩阵和故障概率不等时各概率如表 1 所示,分别验证了其故障概率在相等和不等两种状态下的情况,比较该实例在 AO*, AO_δ* 和 DPSO-AO_δ* 3 种算法下各参数值如表 2、表 3 所示.

表 1 超出差接受器系统故障-测试关联矩阵

故障 状态	测试节点																																				故障发生 概率	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36		
1	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0.00185	
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0.00923	
3	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0.18500	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.00185	
5	0	0	0	1	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	1	0	1	1	0.00185
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0.00923	
7	1	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0.00185	
8	1	0	1	1	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	1	1	0	1	0	0	1	1	0.00923	
9	1	0	1	1	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	1	0	1	0	1	0	0	1	1	0.18500	
10	0	0	0	1	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	1	0	0	0	1	1	0.18500	
11	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	1	0.18500
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.00185	
13	0	0	0	1	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0.00923
14	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0	0	0	1	0	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	0.18500	
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.00923	
16	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.00185
17	0	0	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	1	0.00923
18	1	1	1	1	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	1	0	1	0	0	1	1	0.00185	
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.00185	
20	0	0	0	1	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	1	1	1	0	0	1	1	0.00185	
21	1	0	0	1	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	1	0	0	1	0.00185	
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0.00185	

表 2 故障概率不等时各算法参数性能比较

三种不同算法 性能比较(故障 概率不相等)	算法参数					
	扩展节点数目	回溯的次数	平均测试代价	平均测试节点数	测试代价标准差	运行时间/s
AO* 算法	998	132	3.35	3.35	0.278	4.687
AO 算法	105	3	3.16	3.16	0.223	3.281
DPSO-AO _δ * 算法	43	0	3.02	3.02	0.157	0.265

表 3 故障概率相等时各算法参数性能比较

三种不同算法 性能比较(故障 概率相等)	算法参数					
	扩展节点数目	回溯的次数	平均测试代价	平均测试节点数	测试代价标准差	运行时间/s
AO* 算法	167	1	4.54	4.54	0.498	0.456
AO 算法	85	1	4.17	4.17	0.405	0.296
DPSO-AO* 算法	43	0	4.09	4.09	0.366	0.109

本实例中每一步 AO* 算法的 DPSO 的种群数目为 20,最大迭代次数为 50,DPSO 对根节点的测试集优化结果为

$$T_r = \{t_4, t_8, t_{11}, t_{12}, t_{17}, t_{19}, t_{21}, t_{22}, t_{26}, t_{28}, t_{29}, t_{30}, t_{31}, t_{33}, t_{34}\},$$

DPSO 算法将 AO* 根节点的测试集数目首先从 36 降为 15,从根本上减少了测试个数,本文设每个要扩展节点的 $\delta < 1.2$,将该算法的回溯次数减少到 0,表 1 的实例所得的最优序贯集为

$$\{t_{30}, t_{26}, t_{31}, t_{34}, t_8, t_{11}, t_4, t_{21}, t_{22}, t_{12}, t_{28}, t_8, t_{19}\}.$$

DPSO-AO* 算法将该系统隔离到单个故障状态只需要 13 个测试点,而文献[5]的算法验证了表 1 所示系统(故障发生概率不等),通过遗传算法优选后,所需的平均测试代价(费用)为 3.9526,是本文算法的 1.3 倍.

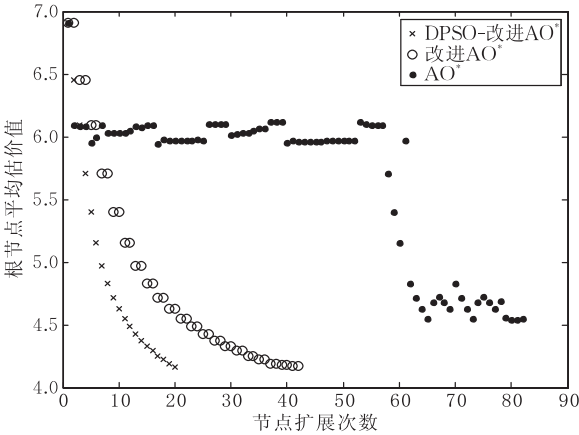


图 1 不等概率条件下三种算法根节点估价值变化趋势

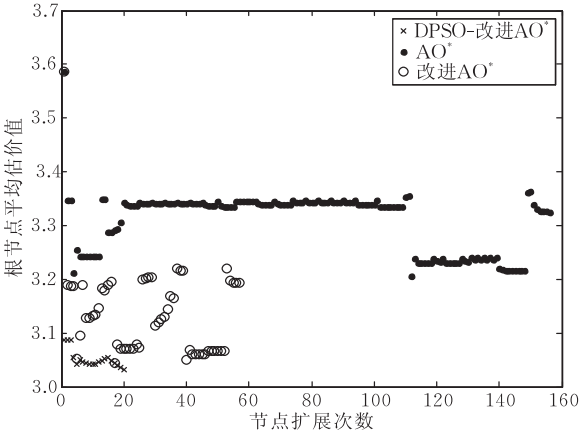


图 2 不等概率条件下三种算法根节点估价值变化趋势

从图 1、图 2 根节点估价值的变化趋势可以看出,该算法效率高的原因在于在整个节点的扩展过程中没有回溯,它所选的每一步测试已经达到最优,所以保证能快速找出最优测试集,且测试代价最低.

从表 2、表 3 可以看出:本文算法扩展的节点数远远小于原有 AO* 算法,测试代价降低了 10%,故障发生概率不等时,测试时间缩短了将近 18 倍;概率相等时,测试时间缩短了 4 倍.

6 各算法性能比较

6.1 计算复杂度分析

文献[6]中采用 bottom-up 算法,在每次迭代中最多有 C_m^2 次序贯测试(采用信息论的启发式算法),最多有 C_m^2 次测试费用估价,而信息论的启发式算法的计算复杂度为 $O(H) = O(mn)$,所以整个 bottom-up 算法的计算复杂度为 $O(m^3O(H))$.

文献[5]中采用遗传算法所用参数为:染色体种群数目 $s = 80$,染色体长度 = 22,交叉概率 = 0.8,变异概率 = 0.01,参与竞争的个体数 = 5.每次进化需计算 S 次适应度及其标定值、选择与变异概率, $S/2$ 次交叉概率,最多需要进行 $sm/2$ 次(m 为系统故障源总数, n 为系统提供的测试点总数目)交叉、变异与逆转,群体更新需要 S 次,所以每次迭代的最坏计算复杂度为 $O(ms^2)$.

本文中的 DPSO-AO* 算法,在每次扩展的节点中,均通过 DPSO 算法进行节点选取,从前面的分析可以看出:对于表 1 所示系统,整个系统要隔离所有故障所需的测点数为 13 个,DPSO 算法通过对 36 个测试点进行一次选择后,选出的测点数为 15 个,基本上可以达到近优(near-optimal),所以 BPSO-AO* 算法在根节点扩展时的数目减少了 21 个,随着节点的扩展,DPSO 每次优选的测点数目呈指数级下降,所以最坏情况下 DPSO-AO* 算法的计算复杂度为 $O(m(n-r))$.

6.2 计算精度分析

文献[6]中,bottom-up 算法的主要贡献就是克

服了 top-down 算法可能找不出全局最优解的缺点,其精度非常高,但其精度的提高是以较长的算法运行时间为代价的,不适用于大型复杂电子系统。

文献[5]中遗传算法寻找最优序贯测试的方法主要是通过算法的多次迭代,种群的交叉、变异等操作完成算法寻优过程,算法最终的寻优精度主要依赖于适应度函数的构造和算法的参数设置,而实际系统中,要用适应度函数完全代表最优序贯测试所要求的所有指标是非常困难的事情。

本文 DPSO-AO* 算法,通过 DPSO 全局寻优算法选取最优的测试点,既减少了扩展节点数目又增加了计算精度,而 AO* 算法通过规定各节点估价值 δ 的范围,当 δ 在规定的范围内进行算法有效的回溯,进一步提高了算法精度。

7 结 论

通过算法性能分析可以看出,DPSO 算法对 AO* 每一个要扩展的节点进行测试选择,将待扩展测试点数目大大降低,克服了原有 AO* 算法^[1]和 bottom-up 算法^[6]不适用于大型复杂系统的缺点;DPSO 与 AO* 算法的结合提高了原有遗传算法^[5]在序贯测试中的运算速度和计算精度。经实例验证表明,DPSO-AO* 算法能够快速找到最优序贯测试集,其测试代价和测试时间明显优于原有 AO* 算法、遗传算法和 bottom-up 算法,有效避免了“计算爆炸”,对于大型复杂电子系统最优序贯测试问题具有很大的参考价值。



JIANG Rong-Hua, born in 1981, Ph.D. candidate. Her research interests include design for testability and artificial intelligence.

Background

The research in this paper is concentrated on optimal test sequencing problem, which is taken account into in design for testability and fault diagnosis field. The existing methods for test sequential problem can easily cause 'computational explosion' and not be fit for large-scale system. This paper is supported in part by the National Defense Basic Research under grant No. A1420061264 and General Armament Department pre-inquest Foundation under grant No. 51317040102. The first grant is for system design for

参 考 文 献

- [1] Pattipati K R, Alexandridis M. Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Transactions on SMC*, 1990, 20(4): 872-887
- [2] Raghavan V, Shakeri M, Pattipati K R. Optimal and near optimal test sequencing algorithms with realistic test models. *IEEE Transactions on SMC*, 1999, 29(1): 11-27
- [3] Shakeri M, Raghavan V, Pattipati K R. Sequential testing algorithms for multiple fault diagnosis. *IEEE Transactions on SMC*, 2000, 30(1): 1-14
- [4] Tu Fang, Pattipati K R, Deb S et al. Computationally efficient algorithms for multiple fault diagnosis in large graph-based systems. *IEEE Transactions on SMC*, 2003, 33(1): 73-85
- [5] Yu Jin-Song, Xu Bo, Li Xing-Shan. Generation of test strategy for sequential fault diagnosis based on genetic algorithms. *Journal System Simulation*, 2004, 16(4): 833-836 (in Chinese)
(于劲松,徐波,李行善. 基于遗传算法的序贯诊断测试策略生成. *系统仿真学报*, 2004, 16(4): 833-836)
- [6] Kundakcioglu O Erhun, Ünlüyurt Tonguç. Bottom-up construction of minimum-cost AND/OR trees for sequential fault diagnosis. *IEEE Transactions on SMC*, 2007, 37(5): 621-629
- [7] Kennedy J, Eberhart R C. Particle swarm optimization//*Proceedings of the IEEE Conference on Neural Networks. IV*, Piscataway, 1995: 1942-1948
- [8] Kennedy J, Eberhart R C. A discrete binary version of the particle swarm algorithm//*Proceedings of the World Multiconference on Systemic, Cybernetics and Informatics*. Piscataway, NJ, 1997: 4104-4109
- [9] Eberhart R C, Shi Y. Comparing inertia weights and constriction factors in particle swarm optimization//*Proceedings of the 2000 Congress on Evolutionary Computation*. San Diego, 2000: 84-88

WANG Hou-Jun, born in 1961, Ph.D., professor, Ph.D. supervisor. His research interests include automatic measurement system, data test, fault diagnosis of military electronic equipment.

LONG Bing, born in 1974, Ph.D., associate professor. His research interests focus on design for testability.

testability analysis, and the second grant is to development an integration software environment for system design for testability. The paper proposes a DPSO-AO* algorithm for test sequential problem, which utilizes the DPSO algorithm to select the test point at each nodes of fault isolation tree and improves AO* algorithm to adjust the test sequence. The proposed method highly reduced the number of created test nodes than AO* algorithm, saved 10% test cost and shorten four times test time.