doi: 10.7690/bgzdh.2014.10.006

一种基于多测试资源的并行测试任务调度算法

姚静波[1]，辛朝军[2]，蔡远文[1]
（1. 装备学院航天装备系，北京 101416；2. 装备学院研究生院，北京 101416）

摘要：为在复杂测试工作中充分利用测试资源、缩短测试时间。通过分析现有并行测试任务调度算法的不足，对 TaskScheduler-T 算法和并行测试完成时间极限定理进行了深入研究，基于测试资源约束，改进了并行测试任务调度数学模型，设计出一种基于测试资源数量约束的并行任务调度改进算法，并以实例对算法进行比较、检验。结果表明，该算法能较大幅度地缩短测试时间、提高测试资源利用率。

关键词：并行测试；任务调度；算法
中图分类号：TP206 **文献标志码：**A

A Task Scheduling Algorithm of Parallel Test Based on Multi Test Sources

Yao Jingbo[1], Xin Chaojun[2], Cai Yuanwen[1]
(1. *Department of Spaceflight Equipment*, *Academy of Equipment*, *Beijing* 101416, *China*;
2. *College of Graduate*, *Academy of Equipment*, *Beijing* 101416, *China*)

Abstract: In order to take full advantage of test resources and reduce test time in complex test tasks. The existing task scheduling algorithms of parallel test were analyzed, and their shortcomings were pointed out. In this article, the TaskScheduler-T algorithm and the limit theorem of the finish time in parallel test have been studied in depth. Base on the restriction of test sources, the mathematic model of parallel test task scheduling was ameliorated, and a new improved algorithm which was compared and tested by example was put forward. The result proves that the new algorithm could greatly reduce the test time and improve the test source utilizing ratio.

Keywords: parallel test; task scheduling; algorithm

0 引言

在自动测试领域，传统的测试方法是对被测单元(unit under test，UUT)逐一测试的串行测试方法，测试时间长，测试设备闲置率高，不利于测试系统效能的充分发挥[1]。并行测试技术是把并行处理技术引入到测试领域所形成的测试方法和技术，通过对测试任务的合理规划来优化利用系统资源，提高测试系统的吞吐率,进而提升测试效率和测试质量，减少测试时间，降低测试成本，是下一代 ATS“NxTest”体系结构中的关键技术之一[2]。

实现并行测试的关键是任务调度[3]。为了实现并行测试任务的最优调度，肖明清课题组提出了多种基于智能算法的并行测试任务调度算法[4]。由于状态方程的某些缺陷，这些算法经常会出现伪解或陷入局部最优解[5]。胡瑜提出了一种 TaskScheduler-T 算法[6]，能够以随机搜索的方法寻找出较优的任务执行序列，但随着测试任务数量的增加，其计算耗时将会急剧增加。方丹根据李云涛提出的并行测试完成时间极限定理提出了 1 种 TSUL 算法[7-8]，基于将稀缺度最高的资源优先分配给需求度最高的测试任务的思路，可获得测试时间最短的最优解，但这种算法的运算量比 TaskScheduler-T 算法更大。上述所有算法均是基于如下假设：1) 所有资源数量均唯一，且同一时刻同一个资源只能被一项测试任务占用；2) 测试任务占用该资源的时间等于任务的测试用时。但在实际测试任务中，测试资源的数量往往是测试过程的重要约束之一；因此，在进行任务调度计算时必须考虑测试资源的实际情况，以充分发挥测试资源的效用。

笔者设计了一种基于测试资源数量约束的并行测试任务调度算法。该算法在设计过程中充分发挥了测试资源的数量特征，以并行测试完成时间极限定理和 TaskScheduler-T 算法为基本依据，生成了在有限测试资源约束下测试时间最优任务序列。

1 任务描述

1.1 相关概念

并行度：设任务集 $T=\{t_1, t_2, t_3,\cdots, t_m\}$的总测试时间为 λ，任务集的基数$|T|$是集合中所有元素出现的总次数，则整个测试过程中单位时间内并行运行

收稿日期：2014-04-24；修回日期：2014-05-19
作者简介：姚静波(1969—)，男，甘肃人，硕士，副教授，从事飞行器测试技术工程研究。

的测试任务数称为该测试任务的平均并行度，简称并行度：

$$\eta=\frac{|T|}{\lambda} \tag{1}$$

当测试任务集确定之后，任务集的并行度越高则总测试时间越短。

资源利用率：仪器资源被测试任务占用的时间与测试任务集的总测试时间之比称为资源利用率。

资源相关：设 t_a、t_b 为 2 个测试任务，R_a 和 R_b 分别是 t_a 和 t_b 占用的测试资源集，若 $R_a \cap R_b \neq \phi$，则称 t_a 和 t_b 资源相关；否，则称之为资源无关。

时序相关：设 t_a、t_b 为 2 个测试任务，若 t_a 和 t_b 两者的测试顺序在测试任务中不能改变，则称这 2 个测试任务时序相关。

串行任务链：在并行测试系统中，由于不同测试任务之间存在的时序相关性或资源相关性，以及在单个 UUT 上进行多个测试任务时，UUT 存在的唯一性等原因，导致这些测试任务只能串行测试，将这些任务组成的集合称为测试资源串行任务链。

并行测试完成时间极限定理：多 UUT 并行测试的完成时间不小于最长串行任务链中所有任务时间之和。其中，最长串行任务链是指任务时间之和最长的串行任务链。

为了便于分析，笔者在算法设计时暂时不考虑并行测试系统中测试任务之间的时序相关性。根据并行测试完成时间极限定理和式 (1) 可知，为了实现测试任务平均并行度最大化，缩短任务集的总测试时间，同时提高测试资源利用率，调度算法必须设法减小最长串行任务链。

1.2　并行测试任务调度数学描述改进

目前研究的并行测试任务调度可以描述为：给定测试任务集 $T=\{t_1, t_2, t_3, \cdots, t_m\}$和仪器资源集 $R=\{r_1, r_2, r_3, \cdots, r_n\}$，已知各项测试任务所需占用的仪器资源、相应的测试用时，以及测试任务之间的时序约束，要求确定并行测试任务调度序列 TS，使得并行测试总时间最短[6]。

基于前面 2 个假设条件的并行测试任务调度问题，可以用以下数学模型描述：

$$\begin{aligned} &F=\min\{\mathrm{time}(TS)\} \\ &\text{s.t.}\begin{cases}\text{if}(t_j \succ t_i),\text{then } N(TS,t_i)>N(TS,t_j) \\ \text{if}\left[N(TS,t_i)=N(TS,t_j)\right],\text{then } R_i\cap R_j=\phi\end{cases}\end{aligned} \tag{2}$$

式中：time(TS)表示任务调度序列 TS 的总测试时间；$N(TS, t_j)$表示任务 t_j 在任务调度序列 TS 中的排序号；R_i 表示任务 t_i 占用的测试资源集。约束条件 1 表示优先级高的测试任务优先测试，约束条件 2 表示同时测试的 2 个测试任务不能资源冲突。

考虑实际测试资源约束条件下，由于某些测试资源的数量超过 1 台，此时约束条件 2 可以表示为：

$$\text{if}\left[N(TS,t_i)=N(TS,t_j)\right],\text{then } R_i\cap R_j\subset \dot{R} \tag{3}$$

式中 $\dot{R}$ 称为盈裕资源集，即任务的测试资源集减去其基集(各测试资源仅出现一次)，剩余的测试资源组成的集合。显然，由于盈裕资源集的存在，原来系统中资源冲突的多个测试任务亦可并行测试，从而有可能使系统的最大串行任务链减小，进而实现总测试时间的缩短和资源利用率的提高。

2　算法设计

基于测试资源的并行测试任务调度算法基本思路：建立任务资源相关矩阵，生成每个资源的任务链，根据测试资源约束条件对最大串行任务链进行拆分，尽可能使系统的最大串行任务链最小，以最终结点数最多的任务链为主链，从其各结点分别生出其他所有任务链，最后生成具有最短测试完成时间的并行任务序列。本算法的流程图如图 1 所示。

对于测试任务集 $T=\{t_1, t_2, t_3, \cdots, t_m\}$和资源集 $R=\{r_1, r_2, r_3, \cdots, r_n\}$，本算法生成任务序列的步骤为：

1) 生成任务资源相关矩阵 $\boldsymbol{TR}^{m\times n}$；

2) 检查相关矩阵中的列向量，删除被覆盖列向量，得到新的任务资源相关矩阵$\boldsymbol{TR}^{m\times n'}$；

3) 根据相关矩阵中各列向量中元素的值，分别得到与各资源相关的任务集 TR_j；

4) 根据任务集中各任务测试时间的长短对任务进行排序，得到任务链 TL_j；

5) 根据任务链测试总时间求出最长任务链，即最长串行任务链 $TL_{\max}$；

6) 根据测试资源情况拆分 $TL_{\max}$，拆分后的次任务链数量等于与之相关资源的数量 N，并使各次任务链总测试时间之差最小，同时从盈裕资源集 $\dot{R}$ 减去该资源；

7) 若 $\dot{R}=\phi$，继续步骤 5)，否则转向步骤 8)；

8) 寻找结点数最多的任务链 $TL'_{\max}$，并将其作为主链向任务序列赋初值，TS:=$\{TL'_{\max}\}$；

9) 依次判断任务序列 TS 中各结点是否属于其他任务链 TL_k，且 $TL_k\notin TS$，若是，进入步骤 10)，否则转向步骤 11)；

10) 从该结点生长任务链 TL_k，TS:=$TS\cup\{TL_k\}$；

11) 若 $TS \neq T$，继续步骤 8)，否则算法终止。

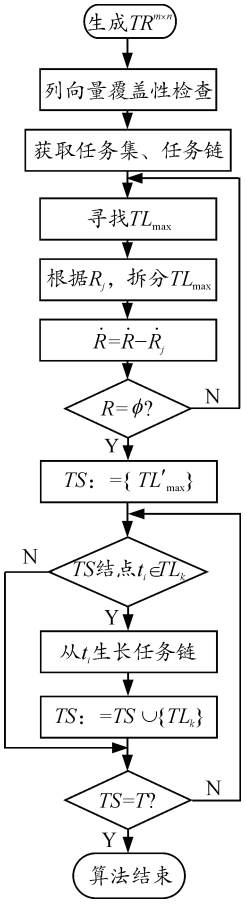

图 1 算法流程

3 算法的比较与检验

3.1 算例描述

对 2 部相同的雷达 A、B 并行测试，单部雷达的测试任务集 $T=\{t_1, t_2, t_3, \cdots, t_8\}$，测试系统的资源集 $R=\{r_1, r_2, r_3, \cdots, r_7\}$，其中，$r_3$ 和 r_7 各有 2 台，因此 $\dot{R}=\{r_3, r_7\}$。根据以往经验数据得到的各项任务的测试时间为 $\{\tau_1$=5 s, τ_2=12 s, τ_3=3 s, τ_4=2 s, τ_5=20 s, τ_6=4 s, τ_7=40 s, τ_8=15 s$\}$，任务资源的占用情况如表 1。所有测试任务之间没有时序相关关系。

表 1 测试任务资源占用情况

测试任务	占用资源	测试时间/s
t_1	r_1, r_2	5
t_2	r_1, r_3	12
t_3	r_4	3
t_4	r_1, r_4	2
t_5	r_3, r_5	20
t_6	r_1, r_6	4
t_7	r_7	40
t_8	r_3	15

3.2 计算结果与分析

根据测试任务建立的相关矩阵为：

$$\boldsymbol{TR}^{8\times 7}=\begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{array}\begin{array}{c} \begin{array}{ccccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{4}$$

显然，r_2、r_6 所在的列向量可被 r_1 所在的列向量覆盖，r_5 所在的列向量可被 r_3 所在的列向量覆盖，根据相关矩阵的列向量可得 2 部雷达并行测试时，相关于资源 r_1，r_3，r_4，r_7 的任务集如下：

$$Tr_1=\{\{t_{A-1}\},\{t_{A-2}\},\{t_{A-4}\},\{t_{A-6}\},\{t_{B-1}\},\{t_{B-2}\},\{t_{B-4}\},\{t_{B-6}\}\}(46)$$

$$Tr_3=\{\{t_{A-2}\},\{t_{A-5}\},\{t_{A-8}\},\{t_{B-2}\},\{t_{B-5}\},\{t_{B-8}\}\}(94)$$

$$Tr_4=\{\{t_{A-3}\},\{t_{A-4}\},\{t_{B-3}\},\{t_{B-4}\}\}(10)$$

$$Tr_7=\{\{t_{A-7}\},\{t_{B-7}\}\}(80)$$

任务集后括号中的数字为该任务集测试总时间，对任务集 Tr_i 中的测试任务按测试时间长短排序即可得相应测试任务链 Tl_i。由并行测试完成时间极限定理可知，该并行测试任务完成的极限最短时间为关于 r_3 的任务链 Tl_3 中所有测试任务的时间之和，即 94 s。由于$|R_3|$=2，因此，任务链 Tl_3 可以拆分为 2 个次任务链：

$$Tl_{3-1}=\{\{t_{A-5}\},\{t_{A-8}\},\{t_{A-2}\}\}(47)$$

$$Tl_{3-2}=\{\{t_{B-5}\},\{t_{B-8}\},\{t_{B-2}\}\}(47)$$

此时，Tl_7 成为最大串行任务链，根据 $\dot{R}$ 对其进行拆分，得到 2 个次任务链：

$$Tl_{7-1}=\{\{t_{A-7}\}\}(40)$$

$$Tl_{7-2}=\{\{t_{B-7}\}\}(40)$$

最终，本次测试的最大串行任务链为 Tl_{3-1} 和 Tl_{3-2}，测试完成时间极限为 47 s，Tl_1 为结点数最多的任务链，以 Tl_1 为主链生成的并行测试任务测试序列如图 2 所示。

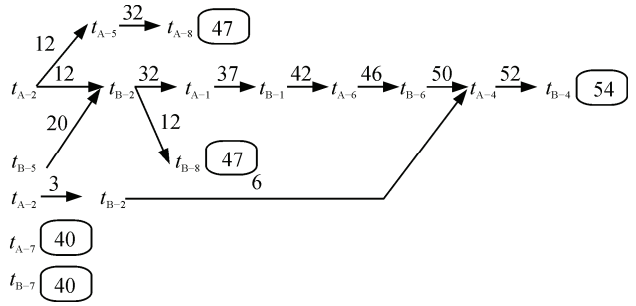

图 2 并行测试任务序列

（下转第 36 页）