文章编号:1671-4598(2023)09-0001-08 DOI:10.16526/j.cnki.11—4762/tp.2023.09.001 中图分类号:TP334.7 文献标识码:A

信息处理单元测试系统并行测试技术研究

周 强,王钰萌,王廷凯
(北京航空航天大学 自动化科学与电气工程学院,北京 100191)

摘要:针对信息处理单元测试的功能需求,设计实现了基于PXIe总线的测试系统,系统包括测控单元、供电单元和接口单元三个组成部分,并完成了测试系统的软件设计;根据信息处理单元测试系统实际的测试流程和所需资源的使用情况,将并行测试技术引入到系统中,重新对相互独立的测试任务进行了并行化分析,构建了该测试系统的任务调度数学模型,提出了一种改进的自适应遗传算法(即IAGA算法),解决了并行测试模块中复杂且难以优化的任务调度问题;对任务调度模型进行了算法仿真验证和实验结果分析,验证了所得解的全局性,所得出的执行序列与串行测试相比,测试效率提高了43.57%,并与其他算法进行了对比,验证了IAGA算法的可行性与优越性;最后将IAGA算法嵌入到了测试系统的软件当中,实现了该优化策略的实际工程应用。

关键词:信息处理单元;PXIe总线;并行测试;IAGA算法

Research on Parallel Test Technology of Information Processing Unit Test System

ZHOU Qiang, WANG Yumeng, WANG Tingkai
(College of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China)

Abstract: Aimed at the functional requirements of information processing unit test, a test system based on PXIe bus was designed and implemented. The system was composed of three parts: measurement and control unit, power supply unit and interface unit, and the software design of the test system was completed. According to the actual test process and required resources of the information processing unit test system, the parallel test technology is introduced into the system, and the parallel analysis of the independent test tasks is carried out again. The task scheduling mathematical model of the test system is constructed to propose an improved adaptive genetic (IAGA) algorithm, solve the complicated task scheduling problem which is difficult to optimize in parallel test module. The algorithm simulation verification and experimental results analysis of the task scheduling model are carried out. Compared with the serial test, the test efficiency of the obtained execution sequence is increased by 43.57%. Compared with other algorithms, the feasibility and superiority of the IAGA algorithm are verified. Finally, the IAGA algorithm is embedded into the test system software to realize a practical engineering application of the optimization strategy.

Keywords: information processing unit; PXIe bus; parallel test; IAGA algorithm

0 引言

信息处理单元是飞控系统的重要组成部分,其主要由计算机板、译码板、遥测板以及总线信号板等部分组成。为了满足质量控制标准,所需验证的测试项目众多,同时每个测试项也需要经过长达数百次的重复测试。常规的测试系统通常采用串行测试的方式,整个测试流程需要的时间较长,测试效率较低。因此,并行测试技术是未来的发展方向。

对于并行测试技术,业内已有部分研究。20世纪80年代,并行处理技术在工业测试领域里开始使用,并逐渐发展成为一种新兴的测试技术——并行测试[1]。并行测试技术将测试流程中的任务重新规划,使占用资源不冲突的任务同时执行,这样能节省大量的测试时间[2]。所以,并行测试技术具有测试速率高、资源利用充分的特点,相较于串行测试,有着巨大优势。目前,美国NI公司在将并行测试技术应用于自动化测试领域最为成功,其开发的智能化测试系统软件平台,将整个测试流程分解为多个子任务,并利用任务调度算法优化求解最优策略[3]。平台中的用户

收稿日期:2023-04-18; **修回日期**:2023-05-23。
作者简介:周 强(1972-),男,博士,副教授,硕士生导师。
通讯作者:王钰萌(1999-),女,硕士研究生。
引用格式:周 强,王钰萌,王廷凯.信息处理单元测试系统并行测试技术研究[J].计算机测量与控制,2023,31(9):1-8,15.

界面简单易使用，用户仅需在人机交互界面中输入待测的测试项和所需时间，测试软件就能快速地生成相应的调度序列，然后将测试调度策略反馈给用户并执行。

根据信息处理单元测试系统实际的测试流程和所需资源的使用情况，将并行测试技术引入到系统中，重新对相互独立的测试任务进行并行化分析。提出改进的自适应遗传算法，即 IAGA 算法，基于 IAGA 算法进行并行测试任务调度算法的设计，并搭建了算法的仿真实验环境，联合本测试系统的任务调度模型得出实验结果并进行详细分析，接着与其他算法进行对比，验证 IAGA 算法的可行性和优越性，并以算法为核心实现了系统中的并行测试模块。

1 信息处理单元测试系统总体设计

信息处理单元测试系统是一套以 PXIe 总线工控计算机为核心的自动测试系统，其具体实现的测试功能包括：①与载机通信测试（1553B、ARINC429）、②一次性指令测试、③遥测功能测试、④RS422 总线通道测试、⑤加速度/角速度脉冲测试[4]。

按照“自顶向下”模块化的设计原则信息处理单元测试系统由测控系统、供电单元、接口单元组成，如图 1 所示。

1.1 测试系统软件实现

信息处理单元测试系统软件基于 Windows 7 操作系统，在 VS2010 平台下采用 C＋＋语言和 MFC 框架进行开发。综合测试软件主要由单项任务测试、板卡自检、并行测试模块、自动测试四个部分组成，综合测试软件界面如图 2 所示。

综合测试软件采用单文档视图结构，利用界面分割技术，实现模块化测试界面的设计，系统的所有测试功能均嵌入在测试界面中，由用户进行选择和配置。板卡影响着

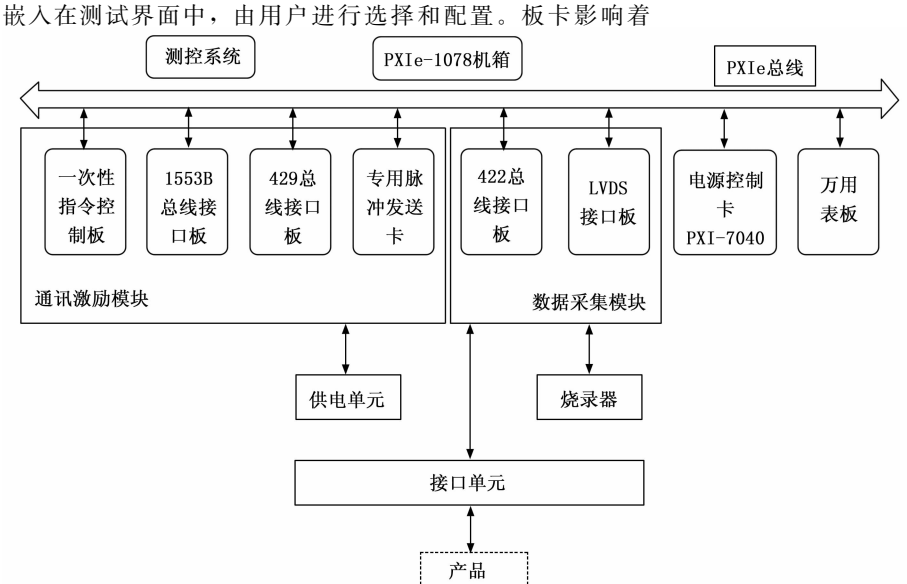

图 1 信息处理单元测试系统总体结构图

图 2 综合软件界面图

测控系统的质量与性能，在用户使用时进行板卡测试十分必要。并且，在自动测试之前，应先设置并行测试模块，不同的状态会有不同的配置选，根据测试的具体情况来输入相应单项测试任务的时间，从而生成并行执行任务序列。

1.2 并行测试模块的实现方案

并行测试模块是信息处理单元测试系统中的关键部分，对系统的测试效率有着重要的影响[5-6]。

图 3 为并行测试模块实现的流程，本文首先对整个系统的测试流程进行分析：飞控产品由飞控计算机部件、译码控制部件、惯测接口部件、总线接口部件四个部分组成，并深入研究这四个部分所涉及的测试任务之间的关系。接下来根据任务所占用的资源情况和任务之间的时序要求建立并行测试的任务调度模型，以得到任务资源矩阵、任务之间的时序约束矩阵等。然后利用改进的自适应遗传算法求解任务调度模型，得到所有可能的并行序列，并通过全局搜索找出最优并行序列，最后将算法嵌入到测试软件中以提供给用户使用。

2 并行测试模块的任务调度模型搭建

2.1 任务调度数学描述

并行测试任务调度问题可以描述为：

测试流程中的任务的集合为 $S=\{s_1, s_2, s_3, \cdots, s_m\}$，测试所需的仪器资源集合为 $W=\{w_1, w_2, w_3, \cdots, w_n\}$，任务相应的占用时长为 $T=\{t_1, t_2, t_3, \cdots, t_m\}$；

所有的测试任务所占用的仪器资源均明确，任务 S_j（$j=1, 2, \cdots, m$）所占用的资源集合为 $W^j=\{w_1^j, w_2^j, w_3^j, \cdots, w_n^j\}$；

令任务资源矩阵为 $\boldsymbol{SW}$，假如任务 S_j 需要其中的资源 W_i，则 $\boldsymbol{SW}(j, i)=1$；否则 $\boldsymbol{SW}(j, i)=0$；

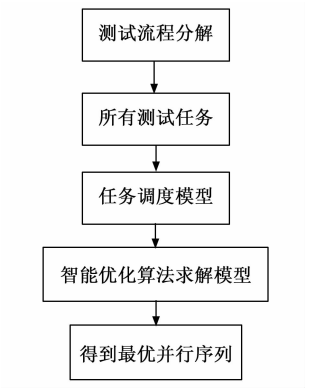

图 3　并行测试实现流程图

令资源相关矩阵为 $\boldsymbol{TW}$，W^A 是任务 S_A 的资源集合，W^B 是任务 S_B 的资源集合，若 $W^A \cap W^B = \neq \varnothing$，则称资源集合 W^A 和资源集合 W^B 相关，$\boldsymbol{TW}$（A，B）$=1$，相当于任务 S_A 与任务 S_B 所需的资源相冲突，不能同时测试；否则，称之为资源不相关，$\boldsymbol{TW}$（A，B）$=0$；

令时序约束矩阵为 $\boldsymbol{TS}$，假如任务 S_A 的优先级（执行顺序）高于任务 S_B，$S_A > S_B$，则 $\boldsymbol{TS}$（A，B）$=1$；否则，$\boldsymbol{TS}$（A，B）$=0$；

令并行测试任务调度序列的矩阵为 $\boldsymbol{T}^P_{n\times step}$，其中 $step$ 表示测试流程中的步数，矩阵中的数值表示某一步可以执行的任务，矩阵中的第 i 列的取值集合表示第 i 步可同时进行的任务集，并且第 i 步所需的测试时长取决于其中所需测试时间最长的任务。

总而言之，并行测试任务调度的数学模型可以概括为：求解任务调度序列矩阵 $\boldsymbol{T}^P$，找出总测试时长最短的 $\boldsymbol{T}^P$，以获得最优的测试效率[7-8]。相当的函数可以表示为 $TF = \min\{sum\_time(T^P)\}$，其中 $sum\_time(T^P)$ 表示任务调度序列矩阵 T^P 执行完成时所需的总时长，此函数的约束条件为：

1）若 $s_j > s_i$，则 $N(\boldsymbol{T}^P, s_j) > N(\boldsymbol{T}^P, s_i)$；

2）若 $N(\boldsymbol{T}^P, s_j) = N(\boldsymbol{T}^P, s_i)$，则 $W_j \cap W_i = \varnothing$。

约束条件表达式中：$N(\boldsymbol{T}^P, s_k)$ 表示任务 s_k 在任务调度序列矩阵 $\boldsymbol{T}^P$ 中的步骤号，1）句表示任务优先级高的测试任务先测试，2）句表示若两个测试任务的步骤号相同，则这两个测试任务没有资源冲突。

2.2　测试流程分析及模型建立

信息处理单元测试系统中的并行测试模块主要由测试任务集、仪器资源以及软件系统三个方面组成。并行测试模块的实现，需要将复杂的测试流程分解为满足时序约束的测试任务集，将被测产品所需的测试通过功能拆分的方式分解为若干项测试任务。信息处理单元测试系统共有 15 项测试任务，依次编号为 $s_1 \sim s_{15}$。根据测试系统的实际测试情况，记录飞控组件中每个测试任务所占用的板卡资源，接着分别记录每个测试任务在常温条件下单独测试时所需要的平均时间。

表 1　常温条件下测试任务所占用的板卡资源与测试时间表

任务/资源	w_1	w_2	w_3	w_4	w_5	w_6	时间/s
s_1	0	1	1	0	0	0	2
s_2	0	0	0	0	0	1	5
s_3	1	1	0	0	0	0	10
s_4	1	0	0	0	0	1	10
s_5	0	1	0	0	0	0	28
s_6	0	0	0	1	0	1	6
s_7	0	1	1	0	0	0	9
s_8	1	0	0	0	1	0	10
s_9	0	0	1	0	1	1	14
s_{10}	0	1	1	0	1	0	14
s_{11}	0	0	0	1	0	0	8
s_{12}	0	1	0	1	0	0	2
s_{13}	0	0	0	1	0	1	6
s_{14}	0	0	1	1	0	0	4
s_{15}	0	0	1	0	0	1	12

根据并行测试的任务调度模型可知，资源任务集矩阵为：

$$\boldsymbol{TW}_{15\times 6} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

根据测试流程中每个测试任务的具体情况可得，每个部分的测试没有必要的时序顺序，但部分内部的测试任务之间有一些时序要求，具体任务之间的时序约束如下：

1）在测试开始前，应首先进行自检测试，即 s_1 优先于 s_2、s_3、s_4、s_5、s_6、s_7、s_8、s_9、s_{10}、s_{11}、s_{12}、s_{13}、s_{14}、s_{15}；

2）在对译码控制部件的测试中，s_3 优先于 s_4；

3）在对惯测接口部件的测试中，s_8 优先于 s_9、s_{10}；

4）在对总线接口部件的测试中，s_{11} 优先于 s_{12}、s_{13}、s_{14}、s_{15}。

那么根据任务调度模型可得时序约束矩阵为：

$$\boldsymbol{TS}_{15\times 15} =$$

$$\begin{bmatrix}
0&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

资源任务矩阵表示每项测试任务所占用的资源数量，时序约束矩阵表示所有测试任务之间的执行先后顺序。因此，两个矩阵中包含某些任务可同时执行的所有信息，通过这些信息可以得出可行的并行序列以实现并行测试。首先，通过资源任务矩阵，提取所有可并行执行的任务集，然后，通过时序约束矩阵将不符合时序约束的任务集进行矫正，具体解决策略采用智能优化算法。

3 任务调度算法设计与并行实现

3.1 遗传算法改进

基本遗传算法收敛速度慢、局部搜索能力差以及容易陷入局部最优解，显然，在解决并行测试任务调度的问题时，需要对其进行改进[9-10]。因此本文提出一种改进的自适应遗传算法，即 IAGA 算法，包含排序分组以及改进的自适应变化的思想，以提高算法的收敛速度，增强算法跳出局部最优的能力[11]。

3.1.1 选择算子的改进

本文采用排序分组（sort-grouped model）选择方式[12]，基本思想是：根据种群中个体适应度值从大到小进行排序，然后将已排序的个体分为三组，通过比例再进行选择，具体实现过程如下：

1）根据个体适应度的大小，对个体进行降序排列。

2）将排序过的种群，按照比例分为：高适应度个体集合（占比 1/8）、中适应度个体集合（占比 6/8），低适应度个体集合（占比 1/8）。

3）将低适应度个体集合直接淘汰，不参与选择，高适应度个体集合按照之前的比例复制两份，中适应度个体集不变。

4）最后将新产生的种群直接全部选择，进入到下一代的遗传进化过程中。

采用这种排序分组的方式，可以将适应度过低的个体直接淘汰，不参与以后的进化过程，从而提升算法的收敛速度，并且通过选择多数适应度中等的个体继续进化，以保留种群的多样性，从而增加找到全局最优解的可能性。

3.1.2 变异算子的改进

对于传统遗传算法来说，变异的概率在进化的过程中是保持不变的，是一个小于 1 的常数值。这种通过变异产生的子代随机性强，会较早的出现“早熟”现象，陷入局部最优的“陷阱”，使得算法失去效力；在进化末期，高度“成熟”的个体依然很可能变异，这会降低算法的收敛的速度，使得进化变得漫无目的[13]。

本文采用一种在进化过程中变异概率跟随个体动态变化的方法，其基本思想是：当个体的适应度大于种群平均适应度时，应尽可能保存当前个体，相应地变异概率应变地很小；反之，个体的适应度越小，为了产生更好的子代，变异概率应变地越大[14-15]。

3.2 算法设计及具体流程

3.2.1 基本数学模型设计

将改进的自适应遗传算法应用于信息处理单元测试系统的并行测试模块中，需要设计一系列可供算法调用的数学模型，包括染色体编码、生成的执行序列矩阵与适应度函数，具体设计如下所示。

1）编码的定义：

将染色体设定为并行测试任务调度扩展矩阵，如公式所示，行号表示板卡资源序号，列号表示测试顺序，每一行代表着占用相应资源的任务集执行顺序，每一列代表着可同时执行的任务集。

$$Chrom^{n\times k}=\begin{bmatrix}
s_{r_1 1}&s_{r_1 2}&s_{r_1 3}&\cdots&s_{r_1 k}\\
s_{r_2 1}&s_{r_2 2}&s_{r_2 3}&\cdots&s_{r_2 k}\\
\cdots&\cdots&\cdots&\cdots&\cdots\\
s_{r_n 1}&s_{r_n 2}&s_{r_n 3}&\cdots&s_{r_n k}
\end{bmatrix}\tag{1}$$

其中：$Chrom^{n\times k}$（r_i，j）表示占用 r_i 号板卡资源的任务号 s_{r_j}在第 j 步执行测试。如果任务集 S 中的某个任务当前被选中，则 $s_{r_j}=s_j$（$s_j\in S$），否则，$s_{r_j}=0$。

2）种群的生成：

首先，从任务集 S 中随机挑选一个任务，通过任务资源矩阵 ***SW*** 确定该任务所需的资源，再将所需资源对应的任务序号放置在染色体中。然后重复之前的步骤，继续选择未被占用资源的任务，直到该步骤执行的任务数量达到饱和，最后继续执行下一步骤的任务选择，直到所有的任务都被选择，具体流程如图 4 所示。

求取并行序列的算法伪代码如下：

```
while S 不为空
k←1;
for j←1 to n //循环找到第 k 步可并执行的序列
if S 不为空//随机选择剩余资源不冲突的任务 例如 s1,s1 占用资源 w1,w4
Chrom(1,k)←1;
Chrom(4,k)←1;
delete(s1,S); //从任务集 S 中删除任务 s1
k←k+1; //当第 k 步任务集达到饱和,继续下一步
```

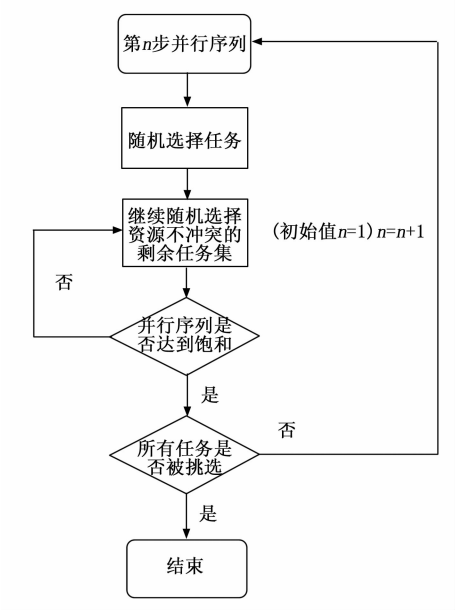

图 4　生成并行序列流程图

3）适应度函数设计：

在遗传算法中，适应度函数的设计十分关键，它直接影响着算法的收敛速度以及算法找到全局最优解的能力。在并行测试领域，如何提升资源利用率和减少测试时间是算法研究的重点问题。因此，并行测试任务调度问题实质上是测试任务与测试资源之间的分配问题，本质上是一个 NP-hard 问题，是多目标组合优化问题。显然，减少总测试时间是第一优化目标，在测试时间最优时，提高资源利用率则是第二优化目标。除此之外，由于产生初始种群时，没有考虑到任务之间的时序性，以至于随机产生的个体有可能是无效解，所以在优化前两个因素之前，保证解的有效性是第一要务。因此，在设计适应度函数时应考虑时序、时间、资源这三方面因素，具体内容如下。

1）错误因子：

错误因子（Error Factor）体现着并行序列中时序的错乱程度，其表达如下：

$$E=\begin{cases}E+10, & s_i\in T^p(,k_1),s_j\in T^p(,k_2)(k_1=k_2)\\ E+2, & s_i\in T^p(,k_1),s_j\in T^p(,k_2)(k_1\neq k_2)\end{cases} \tag{2}$$

由上述公式可知，在调度序列矩阵 $\boldsymbol{T}^p_{n\times step}$ 中，当两个不满足时序要求的任务处在同一列时，换言之它们同时执行，显然这种序列通过进化变成有效解的概率较低，错误因子应该很大；相反，如果这两个任务不同时执行，那么通过遗传算法中的交叉操作很容易变成有效解，错误因子应小些。并且在计算总的错误因子时，应完全遍历调度矩阵中的行和列，找到所有不满足时序约束的任务数。

2）速度比：

速度比（speed ratio）定义为串行测试的总时间与并行序列所需的总测试时间之比，速度比越大，所需测试时间越少。其数学表达式如下：

$$S=\sum_{i=1}^{m}t_i/sumTime(T^p_{n\times step}) \tag{3}$$

其中：$t=\{t_1,t_2,t_3,\cdots,t_m\}$ 表示每个任务单独测试时所需的时长，$sumTime$ 表示求取在并行测试时系统的测试时长的函数，函数的参数为调度序列矩阵 $\boldsymbol{T}^p_{n\times step}$，且每一步执行的时间取决于其中所需测试时间最长的任务。

3）并行率：

并行率（parallel rate）表示在并行测试系统中的资源利用率，并行率越高，资源利用率越高。设定任务调度序列 $T^p_{n\times step}$，中第 k 步并行执行的任务总数为 ξ_k，则表示并行率的公式为：

$$P=\sum_{k}^{step}\xi_k\times 10^{k-1}/(m\times 10^{step-1}) \tag{4}$$

式中，m 表示总任务数，$step$ 表示总步数，其含义是在并行测试的过程中每步执行的任务总数的权重之和与所有任务的权重之比。

综上所述，结合以上三个因素的多目标适应度函数应设计为：

$$f=\frac{1}{1-P+\sigma\times E+\dfrac{\lambda}{S}}=\frac{S}{S\times(1-P)+S\times\sigma\times E+\lambda} \tag{5}$$

式中，σ 表示时序权重系数，λ 表示时间权重系数，这两个参数的大小取决于三个因子对并行测试的重要程度，通过实验分析可知，在进化的前期，$E>10$，$S<2$，故当 σ 取 0.4，取 0.5 时，算法的效果比较好。

3.2.2　算法具体流程

基于改进的改进遗传算法求解并行测试任务调度的算法流程如下：

1）设定该算法的初始参数，包括算法最大的迭代次数、种群数量、交叉概率、变异的最大概率以及最小概率。

2）首先根据种群生成算法初始化种群，初始代数设定为 1。

3）根据多目标适应度函数计算种群中每个个体的适应度大小。

4）基于排序分组的思想，从种群中选出目标个体作为下一代进化过程中的父代。

5）按照设定的交叉概率对被选择的父代个体进行交叉操作。

6）按照设定的变异的最大和最小概率计算出每个子代的变异概率，进行变异操作，并基于禁忌搜索的思想，选择产生的个体。

7）如果迭代次数大于设定值，终止操作；否则，迭代次数自增 1，跳转到步骤 3），继续进行遗传操作。

8）算法结束，选择种群中的最优个体，也就是最优的任务调度序列。

4　算法仿真实验结果与分析

4.1　实验参数设定

选取本系统中的任务调度模型进行算法仿真验证。

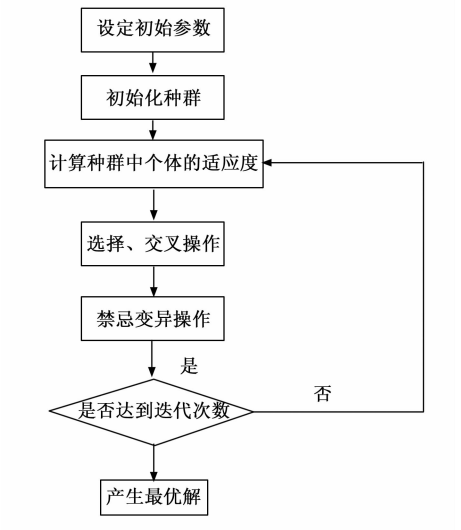

图 5 IAGA 算法流程图

4.1.1 仿真环境配置

仿真实验环境如表 2 所示。

表 2 算法仿真环境配置

序号	类型	名称
1	CPU	Intel(R) Core(TM) i7-10510U
2	机带 RAM	8.00 GB
3	操作系统	Windows 10×64
4	仿真平台	Matlab R2016b

4.1.2 参数设置

经过大量的实验分析，算法的各个参数以表 3 的值时，算法的效果比较好，找到最优解的能力也比较强。

表 3 算法初始参数设置

参数名称	数值	含义
P_C	0.8	交叉概率
$P_{\max}$	0.8	最大变异概率
$P_{\min}$	0.4	最小变异概率
$Popsize$	80	种群规模
$MaxIt$	100	迭代次数
σ	0.4	惩罚系数

4.2 仿真结果及分析

4.2.1 适应度函数收敛曲线

图 6 代表函数的总体适应度与时间变化在迭代的过程中的变化情况。

横轴表示迭代次数，左侧纵轴表示种群总体的平均适应度值，以常数为单位；右侧纵轴表示在每一代种群中所有个体对应的任务调度矩阵所需的平均测试时间，以秒为单位。图中共有两条曲线，实线代表种群在迭代过程中适应度的变化，虚线代表种群在迭代过程中测试时间的变化。由图可知，适应度越高，时间越短，适应度与时间几乎同时变化，两者均在第 15 代开始不再变化，此时适应

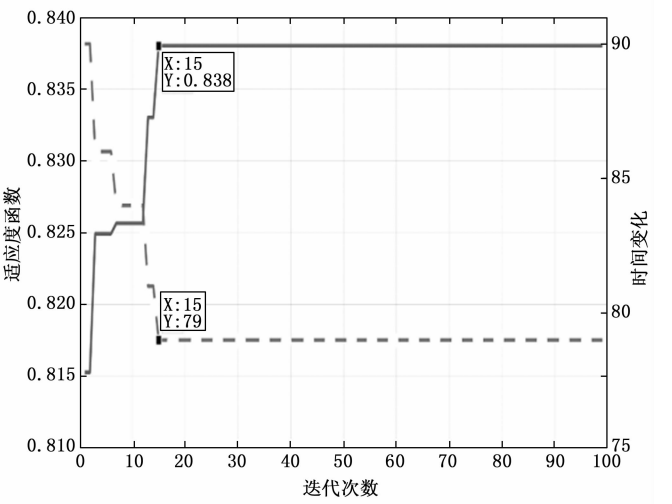

图 6 适应度与时间收敛曲线

度的值为 0.838，测试时间为 79 s。由公式（4）可知，时间因素在适应度函数的权重占比非常高，除错误因子外，是第二重要的影响因素，图中的曲线变化正印证了这一点。

图 7 为函数的总体适应度与并行率变化在迭代的过程中的变化情况。

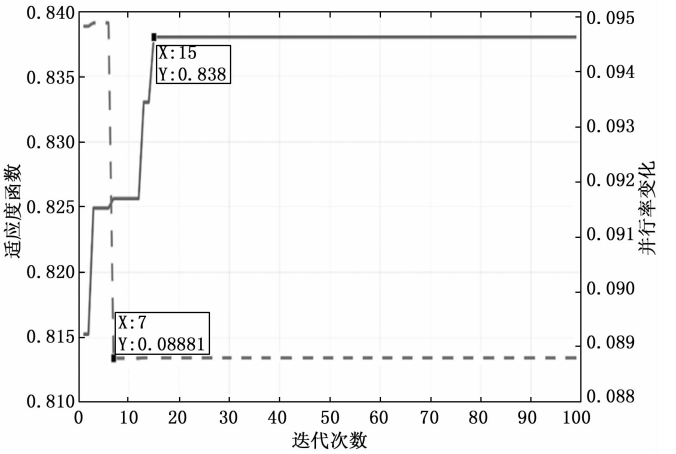

图 7 适应度与并行率变化曲线

横轴表示迭代次数，左侧纵轴表示种群总体的平均适应度值，以常数为单位；右侧纵轴表示在每一代种群中所有个体对应的任务调度矩阵的并行率，以秒为单位。图中共有两条曲线，实线代表种群在迭代过程中适应度的变化，虚线代表种群在迭代过程中并行率的变化。由图可知，适应度的变化与并行率的变化并不同步，并行率高的时候，适应度并不一定高，当两者不再变化时，适应度为 0.838，而并行率为 0.088 8，并不是最高，可见两者的变化相关性较低。由公式（5）可知，并行率在适应度函数的权重占比是最小的，在同时满足错误因子和速度比的优化下，并行率才可能会变高。一定的并行率，即使不是最高，也依然符合优化思路，图中曲线正证明了这一点。

4.2.2　全局最优解的验证

为了验证最优解是否是全局最优，采用 TaskScheduler-T 算法[16]。TaskScheduler-T 算法的基本思想是：随机生成可行的并行测试任务调度，然后枚举所有解得到最优解，它每次都能找到最优解，但相当费时[17]。

图 8 为在样本量为 3 000 的情况下采用 TaskScheduler-T 算法所生成的调度序列对应时间曲线。

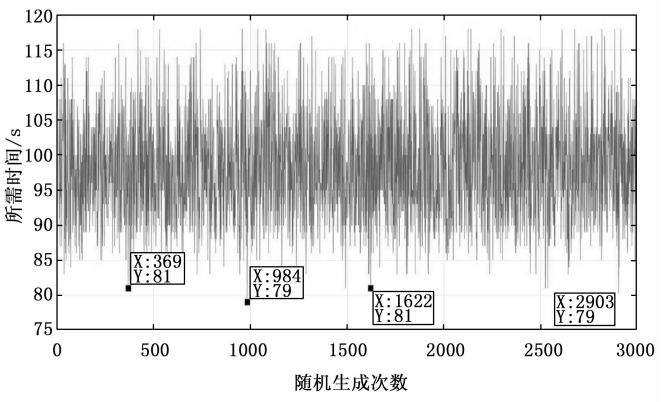

图 8　3 000 次随机生成的样本的测试时间曲线

所有样本中的测试时间集中在 85～115 秒之间，最大值为 118 秒，最小值为 79 秒。可见，在足够多的调度序列中，也未能找到比本论文中采用改进的自适应遗传算法所得到的更好的解。对于 TaskScheduler-T 算法来说，只要样本量足够大，随机搜索的空间也就越大，也就能找出所有可行的解，故本算法所求取的解为全局最优解。

4.2.3　实验结果分析

IAGA 算法得到的最优任务调度序列为 $\begin{bmatrix} 1 & 6 & 5 & 3 & 2 & 4 & 12 & 13 \\ 7 & 9 & 15 & 14 & 10 & & & \\ 8 & 11 & & & & & & \end{bmatrix}$，总测试时间为 79 s，比顺序测试的总测试时间的 140 s 减少了 61 s，提高了 43.57％的测试效率。具体计算公式为：

$$e = \frac{t_s - t_p}{t_s} \times 100\% = 43.57\% \tag{6}$$

其中：t_s 表示顺序测试的总测试时间，t_p 表示并行测试的总测试时间。相应的测试任务在测试资源上的分配的甘特图如图 9 所示。

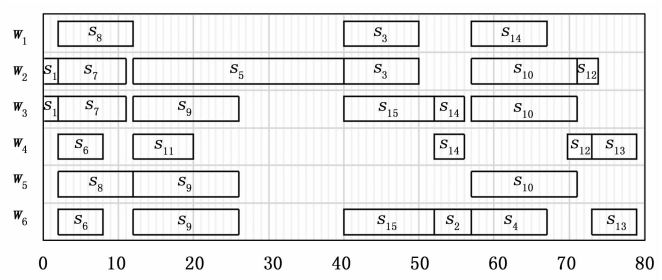

图 9　测试任务执行顺序图

图的横轴表示测试时间，以秒为单位；纵轴表示板卡资源序号，矩形中心表示具体的测试任务，矩形表示相应的执行的过程。它直观地展示了所有测试任务随时间的执行顺序，反映了在整个测试过程中测试任务与板卡资源的占用情况。

表 4 为顺序测试与并行测试在资源利用率上的比较图，其中计算资源利用率的公式为：

$$u_{w_i} = \frac{t_{w_i}}{t} \times 100\% \tag{7}$$

其中：t_{w_i} 为资源 w_i 在测顺序测试或并行测试时工作的总时间，t 表示顺序测试或并行测试的总测试时间。

表 4　顺序测试与并行测试资源利用率的比较

测试方法		顺序测试	并行测试
资源利用率	w_1	21.43％	37.97％
	w_2	44.29％	78.48％
	w_3	39.29％	69.62％
	w_4	18.57％	32.91％
	w_5	27.14％	48.10％
	w_6	37.86％	67.09％
	资源平均利用率	31.43％	55.70％
	总测试时间	140	79

由表 4 可以看出并行测试的资源利用率均比顺序测试的资源利用率高，平均利用率提高了 24.27％。IAGA 算法可以显著提高资源的利用率，从而节约了测试成本。

4.2.4　与其他算法对比

4.2.4.1　与基本遗传算法对比

在实验条件相同的情况下，设置基本遗传算法的变异概率为 0.6，并且其他参数与 IAGA 算法相同，分别运用基本遗传算法与改进的自适应遗传算法（IAGA）求解任务调度。

由图 10 可以看出，基本遗传算法在 38 代左右才找到最优解，而 IAGA 算法在 20 代就找到了最优解。可见，IAGA 算法的收敛速度以及搜索能力均比基本遗传算法好。

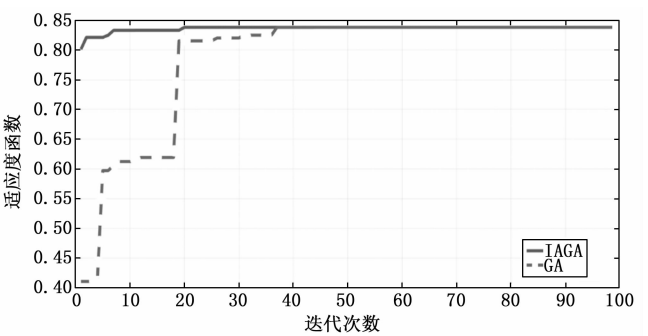

图 10　基本遗传算法与 IAGA 算法对比图

4.2.4.2　与人工蜂群算法等算法对比

人工蜂群算法（artificial bee colony algorithm，简称 ABC 算法）是一种模拟蜂群行动准则的启发式搜索算法，具有收敛速度快、搜索效率高的特点。各工种的蜂群在搜索空间中进行局部寻优行为，找到问题中的可行解，并比

较优劣，逐渐找到全局最优解[18-19]。由于人工蜂群算法与遗传算法同属于启发式搜索算法，理论上，人工蜂群算法的局部寻优能力和收敛速度要强于遗传算法，但全局搜索能力弱于遗传算法，所以采用人工蜂群算法来作比较实验，验证 IAGA 算法的搜索速度以及局部寻优能力[20-21]。

设置人工蜂群算法的采蜜蜂为 40，跟随蜂为 40，侦察蜂为 24，迭代次数限定为 100 次。在实验条件相同的情况下，人工蜂群算法、基本遗传算法、TaskScheduler-T 算法、IAGA 算法均运行 30 次，实验结果如表 5 所示。

表 5 不同算法仿真结果

算法	平均时间	找到最优解的概率
TaskScheduler-T 算法	76 秒	90%
人工蜂群算法	12.842 秒	95%
基本遗传算法	7.0431 秒	90%
IAGA 算法	4.437 秒	96%

由表可得出：TaskScheduler-T 算法一般都能找到全局最优解，但搜索速度太慢，随机性太强。基本遗传算法总体效果均不如 IAGA 算法。在不考虑运行时间的情况下人工蜂群算法与 IAGA 算法的性能相当。通过比较人工蜂群算法的运行时间较长，因为人工蜂群算法在搜索最优解时，速度会逐渐减慢，后期寻优能力较弱。

综合来看，IAGA 算法求解的效果最好，即 IAGA 算法是可行且有效的。可以说明，本算法采用排序分组的思想能使种群往更好的方向进化，采用自适应的变异概率和禁忌搜索手段，可以提高算法收敛速度以及增加种群的多样性，防止算法陷入局部最优。

4.3 并行测试模块实现

飞控产品要在不同的环境下多次测试，测试环境达到十多种，比如高低温测试、不同方向的轴振动测试等。在每个环境下的测试中，每个测试任务执行完成的时间由于受其影响也有所不同。因此，开发一款专门的软件界面提供给用户在不同环境下配置并行测试的执行流程是必要的。系统中的软件可以在不同环境下设置测试任务的时间常数，通过这些时间常数自动生成最佳调度序列，并嵌入自动测试模块中供用户使用。

该软件与信息处理单元测试系统软件界面保持一致性，依旧基于 VS2010 中 MFC 框架开发，而任务调度算法通过 Matlab 平台被编译为动态链接库，以提供给测试软件使用。

用户可以通过单项测试获得不同环境下任务所需的测试时间，并添加在并行测试模块中，生成最佳序列，从而达到根据环境的变化来执行的流程的目的。如图 11 所示，当产品处在高温情况下，生成的调度序列为

$$\begin{bmatrix} 1 & 2 & 5 & 3 & 10 & 12 & 14 \\ & 7 & 9 & 6 & 13 & 15 & 14 \\ & 8 & & & & & \\ & 11 & & & & & \end{bmatrix}$$

，相应的测试效率提高了 41.67%。本软件现已投入使用，证明了本文提出的求解并行测试任务调度问题的 IAGA 算法的工程实用价值。

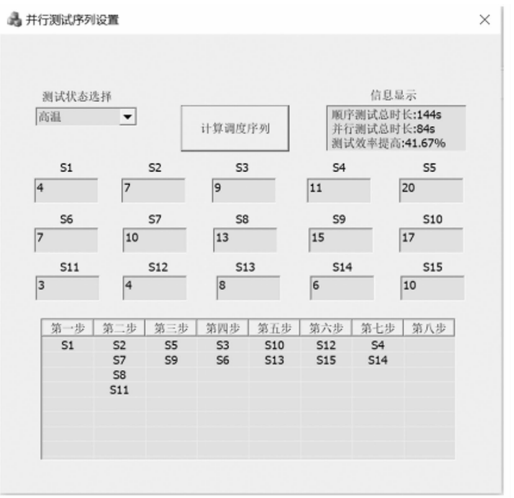

图 11 并行测试模块界面

5 结束语

本文完成了新型信息处理单元测试系统的总体方案设计工作，采用模块化设计思想，将测试系统划分为测控系统、供电单元与接口单元。实现了对系统有着关键影响的并行测试模块，引入并行测试技术，从系统的实际测试环节入手，通过模型中的任务资源矩阵和任务之间的时序约束矩阵，得到测试系统所有可能的调度序列。为了解决普通遗传算法的收敛速度慢，容易陷入局部解的问题，对算法进行了改进，提出 IAGA 算法。使用 IAGA 算法进行了最优任务调度策略求取，成功得到了总测试时间最短、资源利用率最高的调度序列。在常温测试条件下，与串行测试策略相比，采取并行策略的测试系统的测试效率提高了 43.57%。然后与其他算法对比，验证了 IAGA 算法的有效性和优越性。

参考文献：

[1] 王正元，刘卫东，景慧丽，等. 一种并行测试任务调度优化方法 [J]. 兵工学报，2018，39 (2)：399 - 404.
[2] 李铜川，张冠兵，郇黎明，等. 基于 LXI 的机载制导弹药并行测试方法研究 [J]. 计算机测量与控制，2016，24 (7)：144 - 146，151.
[3] 马 敏. 并行多任务自动测试系统分层化建模及其关键技术研究 [D]. 成都：电子科技大学，2008.
[4] SCHWARTZ J. PXI Express technology and mainstream applications [C] // Autotestcon. IEEE，2008：360 - 363.
[5] ALEXEY L. Parallel testing of distributed software [J]. Information and Software Technology，2004，47 (10)：657 - 662.
[6] CHEN P P，QIU B M，BA H. Research on Parallel Test Task Scheduling Based on Improved Genetic Algorithm and Petri Net [J]. Applied Mechanics and Materials，2014，3082 (543 - 547)：1119 - 1122.

(下转第 15 页)