基于混合遗传算法的并行测试任务调度研究

秦 勇 梁 旭
(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:并行测试任务调度核心是将资源合理地分配给测试任务,合理排列任务的执行顺序,最终使整个系统任务执行时间最短。本文提出了一种基于改进的混合遗传算法的并行任务调度算法,并以该算法为基础运用 WPF 和 SQL Server 技术实现了一个任务调度工具软件。算法采用一种结合贪婪算法思想的基因编码方式和交叉变异方法;设计了尺度变换的适应度函数,采用确定式采样选择方法,提高了种群质量。试验结果表明该方法及该工具软件可以有效地解决并行测试任务调度问题。

关键词:并行测试;遗传算法;任务调度;WPF;SQL Server
中图分类号: TP206 TN06 **文献标识码**:A **国家标准学科分类代码**: 510.8060

Research on hybrid genetic algorithm for parallel test task scheduling

Qin Yong Liang Xu
(School of Automatic Science and Electrical Engineering, Beihang University, Beijing 100191, China)

Abstract: One of the key problems to the parallel test task scheduling is distributing resources to tasks and arranging the execution order properly to make sure the test time shortest. A hybrid genetic algorithm and software based on WPF are proposed for the parallel test task scheduling in this paper. The algorithm combines the advantages of genetic algorithm and greedy algorithm, which introduces an improved method for chromosome coding and crossover. And it designs a fitness scaling function and uses the Deterministic Sampling method in order to improve individual quality. The results show that the algorithm and the software can solve the parallel test task scheduling problems effectively.

Keywords: parallel test; genetic algorithm; task scheduling; WPF; SQL Server

1 引 言

自动测试过程中,经常需要对被测件进行几十次甚至上百次的循环测试,因此在自动测试系统的设计过程中,除了系统功能的实现以外,还要考虑如何有效缩短系统执行测试的时间,这也是研究并行测试任务调度意义所在。文献[1]对并行测试技术的发展现状和相关概念做了概括。目前并行测试任务调度问题主要以下有几类解决方法:基于 Petri 网和 Dijkstra 算法的任务调度[2]、基于遗传算法的任务调度[3]、基于粒子群算法的任务调度[4]、基于 DPSO 算法的并行测试任务调度[5]以及蚁群算法与 Petri 网结合的任务调度[6]。这些方法均具有问题描述与建模复杂困难、收敛性能和搜索全局时间最优解能力较低等问题。本文基于以上研究基础,假设任务之间没有时序约束条件,提出了一种基于改进的混合遗传算法的并行任务调度算法,并以此为基础设计实现了简单易用的任务调度优化软件,具有一定工程实践意义。

2 并行测试任务调度问题描述

并行测试任务调度问题可以描述如下[7]:根据测试任务集合 $\boldsymbol{T}=\{t_1, t_2, t_3 \cdots t_n\}$、测试资源集合 $\boldsymbol{R}=\{r1, r2, r3 \cdots r_m\}$、测试任务与资源关系矩阵 $\boldsymbol{TR}^{n\times m}$、测试任务时间集合 $\boldsymbol{Time}=\{\tau_1,\tau_2,\tau_3\cdots\tau_n\}$($\tau_n$ 为任务 t_n 的测试时间)合理安排各个任务的执行顺序,获得最优任务调度序列,使得总测试任务测试时间最短。

3 遗传算法与贪婪算法

3.1 遗传算法

遗传算法是模拟生物在自然环境中的遗传和进化过程而形成的一种自适应全局优化概率搜索算法。它通过

收稿日期:2016-05

对种群进行选择、交叉、变异等操作，逐代将种群进化到包含或接近最优解的状态。遗传算法解决最优化问题的一般步骤如图1。

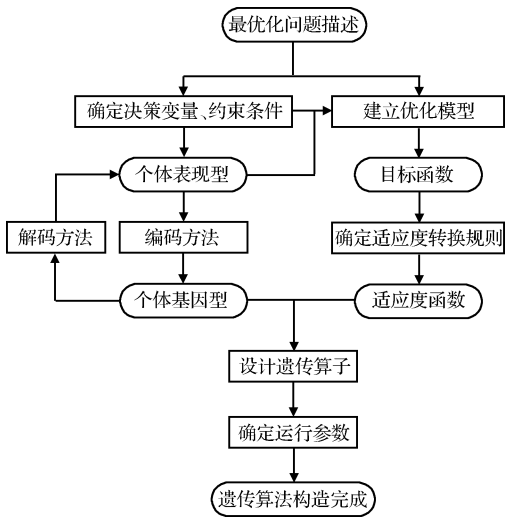

图1 遗传算法的构造过程

3.2 贪婪算法

贪婪算法(greedy algorithm)，又称贪心算法，是一种在每一步选择中都采取在当前状态下最好或最优的选择，从而希望导致结果是最好或最优的算法[8]。但对于大部分的问题，贪婪算法通常得不到最优解，因为该算法没有测试所有可能的解。

4 针对并行任务调度的改进混合遗传算法

并行任务调度具有高复杂度、易死锁、易竞争等特点[9]，所以传统的简单遗传算法选择交叉编译后会产生大量不符合约束条件的低质量个体，大大降低了遗传算法的执行效率以及最优解的质量。本文根据并行任务调度的以上特性，提出了一种结合贪婪算法的改进混合遗传算法。

4.1 染色体编码

根据任务资源关系矩阵 $\boldsymbol{TR}^{n\times m}$ 中被任务调用次数最多的资源的被调用次数，设 $q=s+1$，建立一个 $q\times m$ 的染色体矩阵 $\boldsymbol{X}^{q\times m}$，元素是任务号和任务时间组成的向量，初始值为 $\boldsymbol{v}_0=(0,0)$。每一行表示一个测试任务阶段 Step，前行任务先于后行执行，同一行内任务同时执行。

结合贪婪算法进行染色体编码。逐行进行编码，每一行操作如下：

步骤1：从测试任务集合 $\boldsymbol{T}$ 中随机挑选暂未被分配的任务 t_i，若 t_i 将占用的资源均未被占用，则将该列中 t_i 占用的资源位赋值为向量 $\boldsymbol{v}_i=(t_i,\tau_i)$；若 t_i 将占用的资源已被占，则不执行任何操作。

步骤2：反复执行第一步，直至此行无法再分配任务，此行任务分配停止，开始下一行的任务分配。

最终得到如下染色体：

$$\boldsymbol{X}^{p\times m}=\begin{bmatrix} \boldsymbol{v}_3 & \boldsymbol{v}_0 & \boldsymbol{v}_3 & \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & Step_1 \\ \boldsymbol{v}_0 & \boldsymbol{v}_9 & \boldsymbol{v}_5 & \boldsymbol{v}_5 & \boldsymbol{v}_4 & \cdots & Step_2 \\ \boldsymbol{v}_8 & \boldsymbol{v}_8 & \boldsymbol{v}_0 & \boldsymbol{v}_6 & \boldsymbol{v}_{11} & \cdots & Step_3 \\ \boldsymbol{v}_{16} & \boldsymbol{v}_{10} & \boldsymbol{v}_0 & \boldsymbol{v}_0 & \boldsymbol{v}_{10} & \cdots & Step_4 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & Step_P \end{bmatrix}$$

此方法利用贪婪算法思想，在编码的过程中保证每个Step内的各任务占用资源均不同，因此任务之间不会产生资源冲突，也就有效避免了工作线程间的死锁和饿死等问题。

4.2 适应度函数设计

为了克服遗传算法的早熟现象以及提高后期个体之间的竞争性。本文设计了尺度变换的适应度函数：

$$F(x_i)=t_{\max}-t_{x_i}+1.5\times(t_{\text{avg}}-t_{\min}) \tag{1}$$

式中：$t_{\max}=\max\{t_{x_1},t_{x_2},\cdots,t_{x_m}\}$，为当代个体测试时间最大值；$t_{\min}=\min\{t_{x_1},t_{x_2},\cdots,t_{x_m}\}$，为当代个体测试时间最小值；$t_{\text{avg}}=\sum_{i=1}^{m}t_{x_i}/m$，为当代个体测试时间平均值；$T_{xi}$ 为个体 x_i 的测试时间。

遗传初期阶段 $\Delta t=t_{\text{avg}}-t_{\min}$ 较大，在F中比重较大，从而减小了个体之间F的差异，维护了种群多样性。遗传后期阶段由于个体越来越接近最优解，Δt 较小，Δt 在 F 中比重变小，从而提高了后期个体之间的竞争性。

4.3 交叉与校正

1)交叉算子

考虑并行任务调度染色体编码方式的特殊性，本文采用单点交叉方式。在染色体矩阵 $\boldsymbol{X}^{p\times m}$ 的行中随机选择一行作为交叉点，交叉此行之前的所有行。交叉过程如图2所示。

2)染色体校正

由于交叉操作很容易产生不符合约束条件的子个体，如子染色体 $\boldsymbol{X}_{son2}$ 中 $\boldsymbol{v}_3$ 被分配了两次。所以要对交叉得到的子个体进行校正。

找出子个体中资源冲突任务，将染色体相关位赋值为 $\boldsymbol{v}_0=(0,0)$，然后按编码方式为冲突的任务重新分配资源，最终得到校正后的子个体。

4.4 变异

循环遍历整个染色体矩阵 $\boldsymbol{X}^{p\times m}$，进行变异操作。将变异的任务对应位赋值为 $\boldsymbol{v}_0$，随机选择染色体中任意一行对该任务进行任务分配，并将该行中与变异任务所需资源冲突的任务按编码方式随机分配到其他行。

4.5 选择

为了保证适应度较大的一些个体一定能够被保留在下一代群体中，本文采用了确定式采样选择方法[10]。具体操作过程如下：

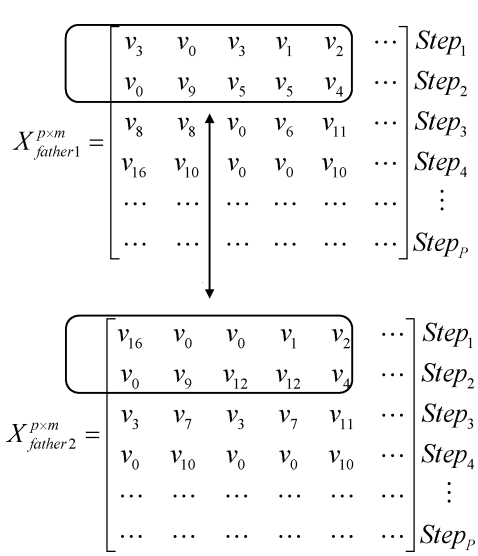

图 2　交叉操作

1)根据个体的适应度计算群体中各个个体在下一代群体中的期望生存数目 N_i：

$$N_i = M \cdot F_i / \sum_{i=1}^{M} F_i \tag{2}$$

式中：M 为种群个体数量。

2)用 n_i 表示不大于 N_i 的最大整数。并将 n_i 暂时作为各个对应个体在下一代种群中的生存数目。

3)经过第二步可以确定 $\sum_{i=1}^{M} n_i$ 个个体，还剩 $M-\sum_{i=1}^{M} n_i$ 个生存指标未分配。按照 N_i 小数部分对个体进行降序排列，并按顺序遍历使对应个体的生存数目 n_i 加1，直到所有生存指标完全分配，即选择出 M 个个体作为下一代种群。

4.6　算法流程

步骤 1：设定各项遗传参数，包括最大遗传代数，交叉概率 p_c，变异概率 p_m。

步骤 2：根据任务资源矩阵 $\boldsymbol{TR}^{n\times m}$ 以及任务时间集合，根据上述编码规则生成初始种群。

步骤 3：评估种群，计算每个个体的适应度 $F(x_i)$。将本代测试时间最短的个体与上一代测试时间最短的个体进行比较，测试时间较短者作为 BI(best individual)，即目前为止最优个体。

步骤 4：采用确定式采样方法从种群中选择 m 个个体作为下一代种群。

步骤 5：以交叉概率 p_c 对本代个体进行交叉操作，并对交叉后的子个体进行校正。

步骤 6：以变异概率 p_m 对个体进行变异操作。

步骤 7：循环执行步骤 3 至步骤 6，直至达到最大遗传代数。取结束时的 BI(best individual)作为最终结果。算法流程图如图 3 所示。

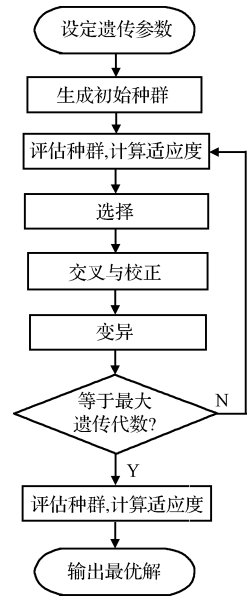

图 3　算法流程

5　并行任务调度工具软件实现及算法仿真

为了验证本算法的性能，本文在 Visual Studio 2013 平台下采用 WPF 技术开发了一个任务调度工具软件，并与 Task-Schedule 算法进行了对比和分析。

5.1　并行任务调度工具软件设计

本软件采用低耦合、可复用以及可独立开发等优点的 MVVM(model-view-viewmodel)编程模式，如图 4 所示。应用 SQL server 数据库进行数据录入、管理和分析，该工具软件具有接口丰富、易扩展易移植、人机交互友好等优点。软件流程如图 5 所示，软件操作界面如图 6 所示。

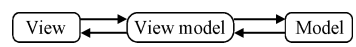

图 4　MVVM 模式

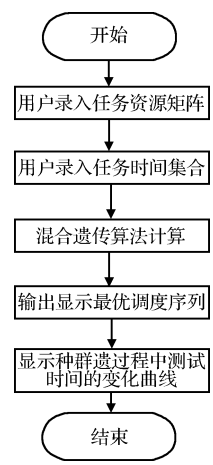

图 5　软件流程

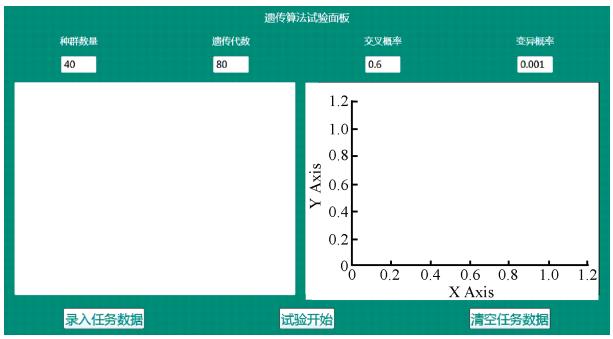

图 6 工具软件操作界面

5.2 算法仿真

已知一个测试任务，测试任务集合 $\boldsymbol{T}=\{t_1, t_2, t_3 \cdots t_{16}\}$、测试资源集合 $\boldsymbol{R}=\{r_1, r_2, r_3 \cdots r_8\}$ 和任务时间集合 $\boldsymbol{Time}=\{30,8,12,14,15,6,20,4,25,10,18,3,16,12,8,10\}$，且假设各任务之间没有前后顺序约束。得到任务资源集合矩阵 $\boldsymbol{TR}^{16\times8}$。

$$\boldsymbol{TR}^{16\times8}=\begin{bmatrix}1&0&0&1&0&0&0&0\\0&1&0&1&0&0&0&0\\1&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&1&0&0&0&1\\0&1&1&0&0&0&0&0\\1&0&0&0&1&0&0&0\\0&1&0&0&1&0&0&1\\0&0&1&0&0&0&1&0\\0&1&0&0&0&1&0&0\\0&1&0&0&0&0&1&0\\0&1&0&0&1&0&0&0\\0&0&0&0&0&0&1&1\\0&0&1&0&0&1&0&0\\0&0&0&0&1&0&0&0\\0&0&1&0&0&1&0&0\end{bmatrix} \tag{3}$$

将 $\boldsymbol{TR}^{16\times8}$ 和 $\boldsymbol{Time}$ 录入到本文实现的基于混合遗传算法的工具软件，遗传参数分别为种群规模 40，遗传代数 200，交叉概率 0.8，变异概率 0.001。最终得到最优个体 X。

$$\boldsymbol{X}=\begin{bmatrix}v_{03}&v_{02}&v_{14}&v_{02}&v_{15}&v_{14}&v_{13}&v_{13}\\v_{07}&v_{11}&v_{16}&v_{05}&v_{07}&v_{16}&v_{11}&v_{05}\\v_{01}&v_{08}&v_{09}&v_{01}&v_{08}&v_{04}&v_{09}&v_{08}\\v_{00}&v_{06}&v_{06}&v_{00}&v_{00}&v_{00}&v_{00}&v_{00}\\v_{00}&v_{12}&v_{00}&v_{00}&v_{12}&v_{00}&v_{00}&v_{00}\\v_{00}&v_{10}&v_{00}&v_{00}&v_{00}&v_{10}&v_{00}&v_{00}\\v_{00}&v_{00}&v_{00}&v_{00}&v_{00}&v_{00}&v_{00}&v_{00}\\v_{00}&v_{00}&v_{00}&v_{00}&v_{00}&v_{00}&v_{00}&v_{00}\end{bmatrix} \tag{4}$$

化简个体 $\boldsymbol{X}$ 可得最优任务调度序列 $Step1=\{t_2, t_3, t_{13}, t_{14}, t_{15}\}$，$Step2=\{t_5, t_7, t_{11}, t_{16}\}$，$Step3=\{t_1, t_4, t_8, t_9\}$，$Step4=\{t_6\}$，$Step5=\{t_{12}\}$，$Step6=\{t_{10}\}$。最终测试时间 85 s，相比传统顺序测试的测试时间 211 s，时间效率提高了 59.7%。按 Task-Schedule 算法由 MATLAB 仿真得到最优序列的测试时间为 86 s，时间效率提高了 59.2%。

对比试验结果可以看出，改进的混合遗传算法得到的最优序列在测试时间上优于 Task-Schedule 算法。由于自动测试系统往往需要长时间循环运行，因此细微的时间优化效用会在循环中逐渐放大，这对于缩短项目周期、节约测试成本异常重要。本算法适合以优化测试时间、提高测试时间效率为最终目标的并行测试任务调度问题。

6 结 论

本文提出了一种基于改进混合遗传算法的并行任务调度方法，并以该算法为基础运用 WPF 技术实现了一个接口丰富、易扩展易移植、人机交互友好的任务调度工具软件。算法采用结合贪婪算法思想的染色体编码方式和交叉变异方法，设计了尺度变换的适应度函数，采用确定式采样选择方法，提高了种群质量。与 Task-Schedule 算法仿真实验对比表明，本文提出的算法能相对有效的求解任务间无前后约束条件的并行任务调度问题。

参 考 文 献

[1] 夏锐，肖明清，朱小平，等. 并行测试技术在自动测试系统中的应用[J]. 计算机测量与控制，2005，13(1)：7-10.
[2] 周强，司丰炜，修言彬. Petri 网结合 Dijkstra 算法的并行测试任务调度方法研究[J]. 电子测量与仪器学报，2015，29(6)：920-927.
[3] 梁旭，李行善，于劲松. 基于遗传算法的并行测试调度算法研究[J]. 电子测量与仪器学报，2009，23(2)：19-24.
[4] 陆晓飞. 基于改进粒子群算法的并行测试任务调度[J]. 信息化研究，2015 (2)：19-22.
[5] 王荣芝，陈晓. 基于 DPSO 算法的并行测试任务调度[J]. 中国测试，2014，40(3)：101-104.
[6] 黄光球，苏海洋，刘冠. 基于蚁群算法的 Petri 网最优路径序列寻找 [J]. 计算机应用，2007，27(4)：932-935.
[7] 袁雪莉，钟明洋. 改进遗传算法的并行任务调度[J]. 计算机工程与应用，2011，47(10).
[8] 方红，杨海蓉. 贪婪算法与压缩感知理论[J]. 自动化学报，2011，37(12)：1413-1421.
[9] 胡瑜. 基于有色 Petri 网理论的并行自动测试系统建模研究 [D]. 成都：电子科技大学，2003.
[10] 杨平，郑金华. 遗传选择算子的比较与研究[J]. 计算机工程与应用，2007，43(15)：59-62.

作 者 简 介

秦勇，1991 年 10 月出生，工程硕士，主要研究方向为控制工程。
E-mail：qinyong1001@126.com

梁旭，1971 年出生，工学博士，讲师，主要研究方向为检测技术与自动化装置。