国 防 科 技 大 学 学 报
第 22 卷 第 3 期 JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY Vol.22 No.3 2000

文章序号：1001-2486（2000）03-0034-05

任务分配与调度中遗传算子的设计*

钟求喜，陈火旺

（国防科技大学计算机学院，湖南 长沙 410073）

摘 要：应用遗传算法等进化方法进行任务分配与调度为越来越多的计算机学者们所关注。基于任务排列的知识表示，常规的标准遗传操作算子并不总是有效的。好的遗传算子对算法收敛性及收敛到好点是非常重要的。在列表编码的知识表示基础上，设计了三个有针对性的遗传算子，即改进的交配算子、内部交配算子和一种作为变异的迁移算子。模拟实验结果与分析表明这些算子对任务分配与调度是有效的。

关键词：任务分配与调度；遗传算法；内部交配；迁移

中图分类号：TP18 文献标识码：A

Genetic Operators in Task Matching and Scheduling

ZHONG Qiu-xi，CHEN Huo-wang

（College of Computer，National Univ. of Defense Technology，Changsha 410073，China）

Abstract：Task matching and scheduling by genetic－algorithm－based approaches have been attractive problems. Standard genetic operators are not always suitable for task matching and scheduling based on permutation representation. Genetic operators are important for genetic algorithms. Three genetic operators are proposed：improved crossover（IMCX），internal crossover（INCX），and migration which transfers a task from a processor to another within a schedule as a kind of mutation. Simulation results and analysis show that these genetic operators are effective for task matching and scheduling.

Key words：task matching & scheduling；genetic algorithms；internal crossover；migration

为能充分利用多处理机系统的并行性，一个应用任务往往被分解成具有数据约束关系的多个子任务。要获得任务的最短完成时间，好的任务分配与调算法是非常重要的。分配是指将多个子任务指派到各处理机；调度是指安排已指派好的各子任务执行顺序。任务分配与调度问题是被公认的 NP 问题[1]。遗传算法是一类模拟生物界自然进化和遗传过程的随机搜索和优化算法，具有在大问题空间求解近似最优解的能力[2,3]。正因遗传算法的强健性和对复杂问题的求解能力，已有学者将其应用于多处理机系统的任务分配与调度[4,5]。遗传算法是基于一个候选解群（解群中候选解的个数称为群体规模）的迭代过程。它使用遗传算子在问题空间进行搜索，采用适应值来评价解群中每一个解的优劣。本文在直观自然知识表示的基础上设计了三个遗传算子，以提高算法的收敛速度及寻找最优解的能力。

1 问题定义

研究任务分配与调度问题时，对问题描述所给出的假设往往是有差别的。不失一般性，假设一个大任务已经被分解成了具有数据约束关系的多个子任务。且系统是非抢先式的，即同一处理机上的一个任务在没有完成前，其它分配到该处理机上的任务不能执行。任务分配与调度问题的一般性描述如下：① n 个子任务的集合 $T = \{T_1, T_2, \cdots, T_n\}$，$T_i$ 为第 i 个子任务；② m 个处理机的集合 $P = \{P_1, P_2, \cdots, P_m\}$，$P_i$ 为第 i 个处理机；③ 一个估计运行时间数组 et[n]，et[i] 表示子任务 Ti 估计运行时间；④ 一个用有向无回路图（DAG）表示各子任务间的数据约束关系的任务依赖关系图。如图 1（a）是有 7 个子任务的依赖关系图。

* 收稿日期：1999-06-11
基金项目：国家自然科学基金资助项目（69903010 和 69783007）
作者简介：钟求喜（1969-），男，博士生。

设 DAG 图 $G = < T, E, et >$，T 为子任务集，一个子任务是 G 中的一个结点，E 是任务依赖关系图的有向边集，et 为估计运行时间数组。$< T_i, T_j > \in E$ 表示子任务 T_i 没有完成之前子任务 T_j 不能执行，称 T_i 为 T_j 的一个前继(PRED)，T_j 为 T_i 的一个后继(SUCC)。任务分配与调度的目标是，在满足依赖关系图的条件下寻找一个分配与调度策略，将 n 个子任务指派到 m 个处理机上并合理调度各子任务的执行顺序，使得整个任务的完成时间最短。

设 S 为 DAG 图的一个分配与调度策略，$t_S(P_i)$ 为策略 S 中处理机 P_i 完成本处理机上最后一个子任务所花费的总时间。设 $t(S)$ 为整个分配与调度 S 的完成时间，有：

$$t(S) = \max_{1 \leqslant i \leqslant m} (t_S(P_i))$$

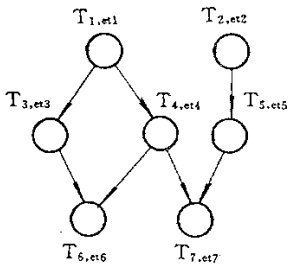

图 1（a） 一个简单 DAG 图

Fig.1（a） A sample DAG

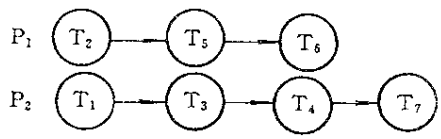

图 1（b） 基于两个处理机的一个合理分配与调度

Fig.1（b） List representation of schedule

目标就是寻找一个分配与调度策略 S，使得 $t(S)$ 最小，即优化目标函数为：$\max_S(t(S))$

2 算法框架

文献［4］较早将遗传算法用于多处理机系统中任务分配与调度，本文称之为标准算法。文献［4］采用的是一种排列方法来表示 DAG 图的分配与调度。每个处理机上所有任务排成一个列表，排列的顺序代表了各个子任务的先后执行顺序。于是 m 个列表就是一个 DAG 图基于 m 个处理机的分配与调度。图 1（b）就是图 1（a）的一个合理分配与调度。合理是指同一处理机上各子任务的执行顺序不能破坏 DAG 图的约束关系。对任务分配与调度问题而言，排列是种自然的表示方法，本文采用这种表示。本文改算法的框架如下：

（1）读入一个 DAG 图，计算图中每个结点的高度值 *height*；

（2）生成初始解群 Pop（t）并计算解群中每个解的适应值，$t = 0$；

（3）如果算法的终止条件不满足，执行步骤 4，否则转步骤 9；

（4）执行选择机制，形成下一代解群 Pop（$t + 1$）；

（5）以概率 Pc 和概率（$1 - Pc$）分别执行内部交配和改进的交配操作；

（6）以概率 Pm 执行迁移操作；

（7）计算解群中每个解的适应值；

（8）采用精英策略保存最优解[6]，$t = t + 1$，转步骤 3；

（9）输出最好解，算法终止。

图中一个结点 T_i 的高度值 *height* 定义如下：

$$height\ (T_i) = \begin{cases} 0，\text{如果 } PRED\ (T_i) = \varnothing，\text{其中 } PRED\ (T_i) \text{ 为子任务 } T_i \text{ 的前继结点集合} \\ 1 + \max\limits_{T_j \in PRED(T_i)} (height\ (T_j))，\text{否则} \end{cases}$$

一个处理机上基于高度值 *height* 的合理调度，是指该任务列表上所有子任务是按高度值升序排列的。如果子任务是按高度值升序排列的，则它是一个合理调度。如图 1（b）中，$height(T_2) = 0 \leqslant height(T_5) = 1 \leqslant height(T_6) = 2$，$height(T_1) = 0 \leqslant height(T_3) = 1 \leqslant height(T_4) = 1 \leqslant height(T_7) = 2$。

定义结点的高度值，一是用于判别分配与调度策略的合理性，二是为初始解群的生成提供依据。初始解群的生成方法是，将 DAG 图中的所有结点集合按高度值划分成 $h+1$ 个子集 DAG(i)，$0 \leqslant i \leqslant h$，$h$ 是 DAG 图的最大高度值。对每个子集 DAG(i)，随机地将子集中的所有子任务指派到 m 个处理机上。然后将所有指派到同一处理机上的子任务按高度值作升序排列得到一个合理的分配与调度。重复执行这个过程就可以生成一定群体规模的初始解群。

适应值用来评价解的优劣。分配与调度策略的完成时间就是一个很好的评价标准。由于遗传算法按比例选择机制中的适应值越大解越好，适应值的计算要对完成时间作一些调整。设 $f(S)$ 表示调度 S 的适应值，则 $f(S) = c - t(S)$，其中 c 是一个足够大的常数以保证 $f(S) \geqslant 0$，如 $c = \sum_{1 \leqslant i \leqslant n} et[i]$。算法中用最多迭代代数以及解群连续一定代数没有进化来作为算法的终止条件。

3 遗传操作算子

3.1 改进的交配算子

文中算法框架中的操作算子与算法[4]是不同的。文献［4］中交配算子对交配点选择的条件要求太强，因而对算子的搜索能力有一定的局限性。对文献［4］中交配算子中交配点的选择作些改进就得到改进的交配算子（IMCX）。对分配与调度策略 S_1 和 S_2 的 IMCX 操作过程如下：

(1) 随机生成一个数 $q(0 \leqslant q \leqslant h)$，在 S_1 和 S_2 的所有任务列表中选择交配点，以便能将每个任务列表分成前后两部分。交配点的选择要满足两个条件，一是所有紧跟交配点后任务的高度值与 q 不等，二是所有紧接交配点前任务的高度值小于或等于 q。

(2) 相互交换 S_1 和 S_2 对应任务列表后半部分所有任务，生成新的分配与调度策略 S'_1 和 S'_2。

由于 IMCX 不会破坏任务列表中任务高度值的排序顺序，因而生成的新分配与调度策略是合理的。

3.2 内部交配算子

交配算子是从两个已知的分配与调度中产生新的分配与调度。但在一个调度内交换不同处理机上的任务也可能产生新的分配与调度策略（内部交配算子，INCX）。对分配与调度策略 S_1 的内部交配操作过程为：

(1) 生成一个任意的随机数 $q(0 \leqslant q \leqslant h)$，从 S_1 中随机选择两个任务列表 L_i 和 L_j，根据 q 在 L_i 和 L_j 中确定交配点以便能分别将任务列表 L_i 和 L_j 分成前后两部分。

(2) 将 L_i 和 L_j 的后半部分所有任务相互交换，生成一个新的调度 S'_1。

定理 1　INCX 中交配点的选择如果满足如下条件：1) 对紧跟交配点后面的任务 T_i，有 $height(T_i) > q$；2) 对紧接交配点的任务 T_i，有 $height(T_i) \leqslant q$，则 INCX 生成的调度也是合理的。

证明　由于 INCX 只发生在一个调度的内，显然各子任务在调度中出现的唯一性可以得到保证。现在只需要证明在交配操作后，各子任务的依赖关系不会被破坏。设交配点在任务列表 L_i 中落在 T_i 和 T_{i+1} 之间，在任务列表 L_j 中落在 T_j 和 T_{j+1} 之间，则有 $height(T_i) \leqslant q < height(T_{i+1})$ 及 $height(T_j) \leqslant q < height(T_{j+1})$。交配操作后，仍有 $height(T_i) \leqslant q < height(T_{j+1})$ 及 $height(T_j) \leqslant q < height(T_{j+1})$。即任务列表中子任务的排列还是满足高度值的升序排列，因而是合理的分配与调度策略。(证毕)

3.3 迁移与变异

遗传算法中的变异操作是一种辅助算子，在解群局部收敛时通过变异算子的突变作用来保持解群一定的多样性。在分配调度问题中，变异操作在一个调度内随机交换两个具有相同高度值的子任务。显然这种交换是合理的。考虑图 2 中的两个调度 S_1 和 S_2。假设某个子任务有三个后继子任务 T_{j-1}，T_j，T_{j+1}，那么调度 S_1 中这三个子任务有可能并行执行，而 S_2 中的三个子任务只能串行执行。这样调度 S_1 就可能比 S_2 有更短的完成时间。图 2 中的串行情况，变异的交换是不可能打破的。为充分利用多处理机的并行性，应允许一个子任务从一个处理机迁移到另一个处理机上。迁移操作在一定程度是能起到变异算子的作用，因为每个子任务都有同样的机会进行迁移。迁移操作的过程为：

(1) 生成一个任意的随机数 q，$0 \leqslant q \leqslant h$。

(2) 计算调度 S 中每个任务列表的高度值等于 q 的子任务数目 N_i，$1 \leqslant i \leqslant m$。

(3) 从最大 $N_i(1 \leqslant i \leqslant m)$ 所在的任务列表中随机选择一个子任务 $T_k(height(T_k) = q)$，将 T_k 迁移到最小 $N_i(1 \leqslant i \leqslant m)$ 所在的任务列表中。

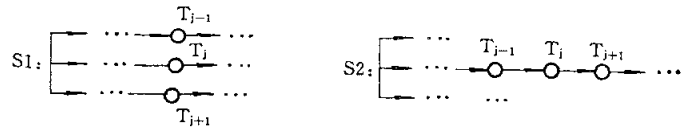

图 2 子任务的并行与串行执行

Fig.2 Parallel tasks and sequential tasks.

一旦确定了要迁移的子任务，下一步是迁移点的选择，以确保迁移后的调度是合理的。迁移算子的合理性由定理 2 来保证。

定理 2 设紧邻插入点前后的子任务分别为 T_i，T_{i+1}，如果 $height(T_i) \leqslant q$ 且 $q \leqslant height(T_{i+1})$，则子任务 $T_k(heigh(T_k) = q)$ 迁移进来后，整个分配与调度仍是合理的。

该定理的证明非常简单，证明过程从略。

4 模拟结果与分析

为了检验文中算子的搜索能力，我们选择文献 [4] 中有代表性数据给出了一些与之比较的模拟结果。算法中主要参数有：群体规模为 10，算法的最多迭代代数为 1000，内部交配概率为 0.8，迁移概率为 0.2。模拟算法 [4] 的交配和变异概率分别为 1.0 和 0.05。DAG 图是的生成与文献 [4] 不同，是任意生成的，每个结点有 1 至 4 个后继，估计运行时间为 1 至 50 间的随机数。运行平台为一台 PII350 的 PC 机。算法能收敛时运行时间通常在 2 秒钟以内。模拟结果见表 1（m 为处理机个数，n 为子任务个数）。由于无法给出最短完成时间，只列出完成关键路径所需的时间作为参考线。图 3 是文中算法基于 4 个处理机分配与调度的静态性能收敛曲线，列出了算法在不同进化代所找到的最优解。

表 1 与算法 [4] 相比较的模拟结果（单位时间）

Tab.1 Simulation resutls (unit time)

m	n	关键路径时间	算法 [4]	文中算法
3	20	240	261	240
	30	315	378	322
	50	511	651	564
4	20	215	249	215
	30	299	333	299
	50	410	476	433

从表 1 中可以看出，文中算法能找到更好的解。图 3 的收敛曲线也表明算法有较好的收敛速度。其中的主要原因，一是内部交配算子与标准交配算子都具有产生好解的能力，二是迁移算子将那些可能串行执行的多个子任务经过程迁移后，有可能获得并行计算的效率。Holland 的模式定理[2] 奠定了遗传算法的理论基础，按比例选择机制将使高出平均适应值的模式在算法的后续代中以指数级方式增加它的试验次数。下面以变异算子为例简单说明迁移算子比文献 [4] 中标准变异算子更有效。

设模式 s 的适应值为 $\bar{f}_s = \sum_{i=1}^{q} f(S_i)/q$，其中 S_1，…，S_q 是当前种群中与模式 s 匹配的 q 个候选解。设存在模式 s'，$\bar{f}_s \leqslant \bar{f}_{s'}$。与模式 s 匹配的个体 S_k 经过变异操作后得到新个体 S'_k。对标准变异操作而言，

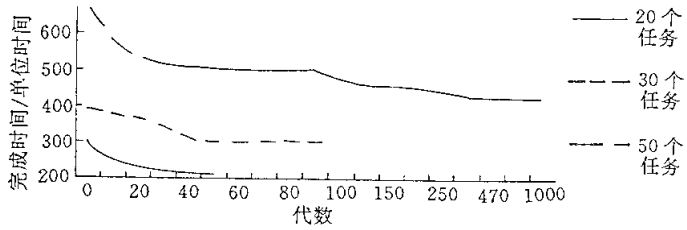

图3 算法收敛曲线（$m=4$）

Fig.3 Finsihing time vs. generations

S'_k 与模式 s' 匹配的概率为 p_s。对迁移操作而言，S'_k 与模式 s' 匹配的概率为 $p_{\text{migration}}$。由于迁移更能充分利用多处理机的并行性，一般有 $p_{\text{migration}} \geqslant p_s$。即迁移算子比标准变异算子更能使算法在后续代中增加高出平均适应值模式的个数。

由于文中算法的交配操作是全概率发生的，因而当问题规模增大时，算法所用时间可能比算法[4]多。但总体结果说明文中设计的遗传算子针对分配与调度问题是有效的。

进一步要考虑的问题是，当问题的假设前提改变后，如考虑处理机间的数据传输延时及处理机处理能力差异等，算法框架和算子有可能要做较大的调整。这正是调度算法从同构系统向异构系统过渡中要解决的问题。

参考文献：

[1] Hironori Kasahara, Seinosuke Narita. Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing [J]. IEEE trans. on Copmuters, 1984, C-33 (11).

[2] John H Holland. Adaptation in Natural and Artificial Systems [M]. Univ. of Michigan Press, 1975.

[3] David E Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning [M]. Addison-Wesley Publishing Company, Inc, 1989.

[4] Edwin S H Hou, Nirwan Ansari, Genetic Algorithm for Multiprocessor Scheduling [J]. IEEE trans. on parallel and distributed systems, 1994 5 (2).

[5] Imtiaz Ahmad, Muhammad K Dhodhi. Multiprocessor Scheduling in a Genetic Paradigm [J]. parallel computing, 1996, (22): 395-406.

[6] Zhong Qiu-Xi, Xie Tao, Chen Huo-Wang. Survival Strategy of solution in GA, Computer Engineering & Science [J]. 2000, 1.