多处理机调度问题的蚁群优化算法

邓 韶^a, 谢晓兰, 程小辉^a

(桂林理工大学 a. 信息科学与工程学院; b. 广西空间信息与测绘重点实验室, 广西 桂林 541004)

摘要: 传统蚁群算法存在收敛速度慢、计算时间长、易陷入局部最优解等方面的缺陷。通过对蚁群信息素更新、策略选择、参数选择等各方面进行改进,提出一种更加高效的多处理机调度蚁群优化算法。实验证明:与其他优化算法相比,该算法能在较短的时间内找到更好的调度策略,具有较好的收敛性和有效性及优良的全局优化性能。

关键词: 蚁群算法; 多处理机调度; 优化; 仿真

中图分类号: TP391

文献标志码: A

0 引言

蚁群算法最早由意大利学者 Dorigo 等^[1]于 20 世纪 90 年代初提出,它是模拟自然界中真实蚁群的觅食行为而形成的一种模拟进化算法。蚁群算法具有较强的鲁棒性、正反馈、分布式计算和贪婪的启发式搜索等优点^[2]。蚁群算法被广泛用来求解二次分配问题、车辆路径规划问题、车间作业调度问题等 NP 完全问题,分布式多处理机系统中任务调度就是一个 NP 完全问题。

目前,针对处理机调度问题的研究成果有:文献[3]提出了一种新颖的蚁群优化算法,用于解决流水作业的优化调度问题,使用结点或弧模式的有向图表示流水作业调度问题,蚁群间的通信机制受有向图上信息素更新过程的约束,最终引导整个蚁群收敛到问题的优化解,但算法并没有考虑收敛时间的快慢问题;文献[4]的思想是为每只蚂蚁分配一个处理机,在每次迭代中,由主控进程生成并行蚂蚁,当所有蚂蚁完成一次搜索时再进行同步,这样虽然每个蚂蚁的计算量很小,但大部分时间却耗在了与主控进程的同步及信息交互方面,增加了算法的执行时间;文献[5]中考虑了一

种最简单的并行方法,即并行的蚁群在求解过程中不进行任何信息交流,它实现了并行的独立运行,计算结果也体现了这一策略的高效性,但是,缺少了蚁群间的信息交流,会使算法极易陷入局部最优解,导致算法停滞等。

通过以上分析可知,目前对利用基本蚁群算法来解决多处理机任务调度问题的研究虽然很多,但是大多数都是针对某一方面的,而且存在各种问题,缺少完整的系统解决方案。因此,本文采用基于一系列改进策略的优化蚁群算法来解决多处理机任务调度问题。在基本蚁群算法的基础上,通过采用动态信息素更新规则、最优策略和参数选择等改进方法,可以在保证收敛速度的条件下较快地得到全局最优解,很好的解决了多处理机任务调度问题。仿真实验证明,该算法在速度、稳定性等方面均优于同类算法。

1 多处理机调度问题的描述

有 m 台相同的处理机($p_1, p_2, p_3, \dots, p_m$) 处理 n 个独立的任务($j_1, j_2, j_3, \dots, j_n$),多处理机调度就是这些任务在满足一定的优先约束关系的前提下,合理分配到各处理机上有序执行,以达到总的执行时

收稿日期: 2012-12-27

基金项目: 国家高技术研究发展计划重大专项(2013AA12A402); 国家自然科学基金项目(61262075/F020702); 广西空间信息与测绘重点实验室开放基金项目(桂科能 1103108-25; 桂科能 1207115-13)

作者简介: 邓 韶(1979—),男,硕士,讲师,研究方向:软件工程, gutming@126.com。

引文格式: 邓韶,谢晓兰,程小辉. 多处理机调度问题的蚁群优化算法[J]. 桂林理工大学学报, 2013, 33(2): 329-332.

间最短,由此假设:

①每台处理机同时只能处理一个任务;

②一个任务不能同时在不同的处理机上进行处理;

③任务一旦开始进行就不能中断;

④所有处理机都是相同的,即每个任务都可以在任一处理机上处理,而且处理时间是相同的;

⑤任务之间存在着一定的优先约束,它规定了任务执行的有效顺序,不允许有循环的优先关系。

处理机调度问题属于离散优化问题,而蚁群算法在求解离散优化问题上的卓越性能为使用蚁群算法求解处理机调度问题提供了可行性。

2 蚁群算法基本原理

仿生学家长期研究发现:蚂蚁视觉不佳,但运动时会通过在路径上释放出一种特殊的分泌物——信息素来寻找路径。当它们碰到一个还没有走过的路口时,就随机地挑选一条路径前行,同时释放出与路径长度有关的信息素。蚂蚁走的路径越长,则释放的信息量越小。当后来的蚂蚁再次碰到这个路口的时候,选择信息量较大路径的概率相对较大,这样就形成一个正反馈机制。最优路径上的信息量越来越大,而其他路径上的信息量却会随着时间的流逝而逐渐消减,最终整个蚁群会找出最优路径。同时蚁群还能够适应环境的变化,当蚁群的运动路径上突然出现障碍物时,蚂蚁也能很快的重新找到最优路径。可见,在整个寻径过程中,虽然单只蚂蚁的选择能力有限,但是通过信息素的作用使整个蚁群行为具有非常高的自组织性,蚂蚁之间交换着路径信息,最终通过蚁群的集体自催化行为找出最优路径。

蚁群算法是一种随机搜索算法,在此算法中,问题的解被抽象成在离散状态空间中从起始状态到达目标状态的状态转移序列,问题的最优解对应着满足最优评价准则的状态转移序列。路径上的信息素强度值是蚂蚁进行状态转移的依据,群体中每只蚂蚁完成一次搜索之后,根据其各自经历的路径来更新路径上的信息素强度,从而完成群体的一次搜索。蚂蚁群体的搜索过程不断循环,最终使最优转移序列上的路径强度达到最大,从而通过蚂蚁个体之间信息的交流与相互协作求得

最优解。

蚁群算法的数学模型具体描述如下:

首先,将 m 个蚂蚁随机的放置在 n 个城市上,城市间的每一路径上都有初始化信息素 $\tau_{ij}(0)$,每个蚂蚁有个状态序列的记录表 $Tabu_k$,用于记录蚂蚁走过的城市,然后,每只蚂蚁根据状态转移概率随机进行状态转移,并且每次转移只允许从当前状态转移到相邻的某个状态,所以,蚂蚁 k 从状态 S_i 转移到某个相邻状态 S_j 的状态转移概率 $P_{ij}^k(t)$ 定义为

$$P_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}{\sum_{S_j \in allowed_k} [\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}(t)]^\beta}, & S_j \in allowed_k; \\ 0, & otherwise. \end{cases} \quad (1)$$

其中: $\tau_{ij}(t)$ 为城市 i 到城市 j 路径上的信息素强度; $\eta_{ij}(t)$ 为城市 i 到城市 j 的距离因子; α 为路径上信息素的重要程度; β 为启发因子的重要程度,且 α 和 β 都大于0; $allowed_k$ 为蚂蚁 k 所允许经过的状态 S_j 的集合。当循环结束后,进行信息素更新。

信息素更新公式为

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{k=1}^m \Delta\tau_{ij}^k(t). \quad (2)$$

其中: ρ 为信息素消散等级常数; m 为蚁群中蚂蚁个数; $\sum_{k=1}^m \Delta\tau_{ij}^k(t)$ 为某条路径上新增信息素的和; $\Delta\tau_{ij}^k(t)$ 为 t 时刻蚂蚁 k 在城市 i 和城市 j 之间路径上的信息素数量,其计算公式为

$$\Delta\tau_{ij}^k(t) = \begin{cases} Q/L^k(t), & S_i \rightarrow S_j; \\ 0, & other. \end{cases} \quad (3)$$

其中: Q 为常数, $L^k(t)$ 为 t 时刻蚂蚁 k 经过路径的长度。

路径长度越短,路径信息素越强,也就会吸引更多蚂蚁向此路径移动,这表明该路径是一个更好的路径。这样,蚂蚁之间可以相互通信、协同工作,基本蚁群算法就是采用这种反复记忆和学习的过程,最后得到全局最优解。

3 多处理机调度中蚁群算法的改进

3.1 信息素更新策略改进

信息素的更新策略是决定收敛速度的关键之一。在蚁群算法中,当蚂蚁从一个具有很多城市

的复杂地图的边部城市出发时,它在出发点附近的局部区域走的路径往往是较优的,走到地图中心区,路径错综复杂,与其他蚂蚁走的路径重叠增多、互相影响加大,走出较优路径的概率降低。因此,为了避免中心区信息素堆积过多,采用了信息素动态更新策略,规定每只从边部城市出发的蚂蚁留下的信息素应随着向中心区的延伸而逐渐减弱,当走到最远处时,其信息素应减到最小。

为此,当每只蚂蚁走完一个边以后,即按下式进行局部信息更新:

$$\Delta\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}, \quad (3)$$

$$\Delta\tau_{ij} = Q/L_z, \quad (4)$$

$$L_z = \sum_{k=1}^m \sum_{l=1}^x d_l^k. \quad (5)$$

其中: Q 为一个较大的常数; x 是蚂蚁 k 在本次循环中已走过的城市边数; d_l 是蚂蚁 k 已走过的边的长; m 是蚂蚁总数; L_z 就是所有蚂蚁在本次循环中已走过的边的累加总长。该动态更新策略保证了各蚂蚁所留信息素的均衡性,保证了对搜索的均衡贡献和相互协作,体现出了群体的力量,能够大幅度地提高算法的收敛速度。

3.2 自适应挥发系数调整方法

信息素挥发因子 ρ 直接关系到蚁群算法的全局搜索能力及收敛速度,由于 ρ 的存在,使那些一直没有被搜索到的节点上信息素会减少到接近于0,降低了算法的全局搜索能力。当 ρ 过小时,以前搜索过的解被选择的可能性过大,会影响算法的全局搜索能力。当 ρ 较大时,虽然可以提高算法的全局搜索能力,但又会降低算法的收敛速度。因此,本文提出自适应地改变 ρ 的值,设其初值为1。在 N 次循环内,当最优值没有明显改变时,

$$\rho = \begin{cases} \rho(t-1), & \rho \leq \rho_{\max}; \\ \rho_{\max}, & \text{else.} \end{cases} \quad (5)$$

式中: $\rho_{\max}$ 为信息素挥发因子最大值。这样可以降低算法的收敛速度,提高搜索最优解的质量。

3.3 算法参数的选择改进

关于蚁群算法中蚂蚁数量 m 的选择,也应该综合考虑算法的全局搜索能力和收敛速度。 m 个蚂蚁在一次循环中所经过的路径,表现为问题集中的一个子集。子集越大(即蚁群数量越多)就越能提高蚁群算法的全局搜索能力以及算法的稳定性,但当蚂蚁数目增大后,信息正反馈的作用

不明显,收敛速度减慢;反之,子集较小,当处理的问题规模比较大时,会使那些从来未被搜索到的解上的信息量减少到接近于0,搜索的随机性减弱,算法的全局性能降低,容易出现早熟收敛。

针对具体问题的应用条件和实际要求,在全局搜索能力和收敛速度两方面作出合理的选择。结合仿真实验,笔者认为,在蚁群算法中蚂蚁数量 m 可取 $[\sqrt{n}, n/2]$ 之间(其中 n 为问题的规模)的一个整数,这样,算法的各项性能相对较为平稳。

4 算法求解过程分析

通过对基本蚁群算法的改进,结合前文的改进策略,对蚁群算法实现多处理机调度的具体求解过程如下:

1) 初始化各项参数值,用一个较小的值 τ_0 设置每条城市间路径上的信息素强度 τ_{ab}^k 和蚁群的最大循环次数,给每个处理机单独分配一个蚁群,并把其他各蚁群中的每只蚂蚁随机放置在任意一个城市上。

2) 系统循环开始,每只蚂蚁按照条件转移概率公式选择下一个应该到达的城市,并且根据局部更新公式进行路径上信息素的局部更新,当所有的蚂蚁完成一次循环后,各处理机蚁群之间进行信息交互,结合所有城市间最优路径上的信息素进行全局更新,得出本次循环的全局最优解。

3) 比较这次和上次的循环完成时间 T ,若优于上次循环时间,保留这次最优路径;否则,继续使用上次最优路径,然后进行下一次循环。

4) 当循环次数大于设定次数,停止循环,最后得到最短循环时间和全局最优解,算法结束。

5 实验仿真及结果分析

5.1 仿真环境及实验参数

为了验证算法的有效性,建立了仿真环境进行验证分析。本实验中,蚁群算法的参数设置为: $Q = 100$, $m = 50$, α 、 β 的初始值为 $\alpha = 1$ 、 $\beta = 4$,随着实验的进行, ρ 值再按照式(5)进行自适应挥发系数动态调整。

在本仿真实验中,采用文献[5-6]中的数据来测试本文算法,通过各算法实验结果的对比分析,验证本文算法的优越性。

实验数据如下：设有3台处理机和9个作业，作业需要运行的时间分别为：81、40、26、4、65、98、53、71和15。采用遗传算法、基本蚁群算法、文献[6]中改进蚁群算法以及本文算法($\beta = 0.8, Q = 100$)，循环50次，得到的结果如表1所示，最优解为 $P_1(98, 53)$ ， $P_2(81, 40, 26, 4)$ ， $P_3(71, 65, 15)$ ，完成时间为151。

表1 实验结果比较
Table 1 Results from experiments

算法	结果比较			
	平均时间/s	平均值	最优解	最差解
遗传算法	0.54	165.2	151	177
基本蚁群算法	0.14	163.6	151	167
文献[6]中的改进算法	0.09	158.3	151	163
本文算法	0.08	155.8	151	161

5.2 仿真实验结果及分析

可以看到，在仿真实验中比较的各项指标，本文算法都比其他几个算法在平均时间、平均值、最优解等方面有所提高，尤其在平均时间的比较上，本文算法能快速的收敛到全局最优解，并得出最优分配调度相对应的最短时间；在算法的平均值上，本文算法也是最接近最优解的；在最差解上，和其他算法相比较，它也是最低值。

另外，仿真实验中测试了本文算法和其他传统算法在各种参数设置同等情况下进行计算时，各次迭代最优结果的进化情况，由于本文算法可使蚁群的搜索向适应度较好的方向进化，因而可以较快地找到较优解，避免早熟现象，使解具有

较好的全局特点。因此，本文算法具有较强的发现最优解的能力，并且从实验的平均时间得知本文算法有更快的进化速度，说明它具有较好的收敛性，也更加有效。

综上所述，对于求解更大规模的处理机调度问题，本文算法是完全有效的。

6 结束语

本文对传统蚁群算法进行了改进，并把该改进算法成功运用到多处理机的任务调度问题中，收到了良好的效果。然而，对于蚁群算法在多处理机的任务调度问题中的研究，也只是做了初步优化尝试，对于如何改进和优化基于多处理机的蚁群算法以获取更好的时间性能，以及考虑测试时间的不确定性等问题，有待进一步进行研究。

参考文献：

- [1] Dorigo M, Maniezzo V, Colomi A. The ant system: Optimization by a colony of cooperating agents [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 1996, 26 (1): 1-13.
- [2] 黎卫红, 刘羽, 芦奉良. 面向多核的时间帧加权公平调度算法 [J]. 桂林理工大学学报, 2012, 32 (1): 121-125.
- [3] 陈晶, 潘全科. 改进蚁群算法求解同型机任务调度问题 [J]. 计算机工程与应用, 2011, 47 (6): 9-16.
- [4] 郭乘涛, 江志斌. 应用混合蚁群算法求解并行批处理机组批与调度问题 [J]. 上海交通大学学报, 2010, 44 (8): 1068-1073.
- [5] 高尚, 钟娟, 莫述军. 多处理机调度问题的蚁群算法 [J]. 微型电脑应用, 2003, 19 (4): 9-16.
- [6] 张勇, 张曦煌. 改进型蚁群算法的多处理机任务调度研究 [J]. 计算机工程与应用, 2007, 43 (35): 74-76.

Multiprocessor Scheduling Based on Ant Colony Optimization Algorithm

DENG Ming^a, XIE Xiao-lan, CHENG Xiao-hui^a

(a. College of Information Science and Engineering; b. Guangxi Key Laboratory of Spatial Information and Geomatics, Guilin University of Technology, Guilin 541004, China)

Abstract: Traditional ant colony algorithm is improved to overcome the limitation of stagnation, slow rate of convergence and long time of computing. A more efficient ant colony optimization algorithm is proposed with multiprocessor scheduling. The algorithm improves ant colony pheromone updates, strategic selection, parameter selection and so forth. Better scheduling strategy is found in short time, with excellent global optimization properties. Simulation results show that this algorithm has better convergence and effective comparison than other optimization algorithms.

Key words: ant colony algorithm (ACA); multiprocessor scheduling; optimization; simulation