研究简报

doi: 10.13413/j.cnki.jdxblxb.2014.03.29

离散粒子群优化算法在流水作业调度问题中的应用

付志军[1], 冯 丽[2], 杜伟宁[3], 凌振宝[1], 杨凤芹[4]

(1. 吉林大学 仪器科学与电气工程学院, 长春 130061; 2. 安阳师范学院 数学与统计学院, 河南 安阳 455002;
3. 空军航空大学 飞行训练基地, 长春 130062; 4. 东北师范大学 计算机科学与信息技术学院, 长春 130117)

摘要: 通过引入随机向量, 改进离散粒子群算法 DPSO 的更新方程, 提出一种离散的粒子群优化算法 MDPSO, 并将其应用于调度问题的求解. 实验结果表明, 该算法优于传统的时序分解算法和遗传算法.

关键词: 进化算法; 粒子群优化算法; 调度问题

中图分类号: TP181 **文献标志码**: A **文章编号**: 1671-5489(2014)03-0561-04

Applications of Discrete Particle Swarm Optimization in Solving the Job-Shop Scheduling Problems

FU Zhijun[1], FENG Li[2], DU Weining[3], LING Zhenbao[1], YANG Fengqin[4]

(1. *College of Instrumentation & Electrical Engineering, Jilin University, Changchun* 130061, *China*;
2. *School of Mathematics and Statistic, Anyang Normal University, Anyang* 455002, *Henan Province, China*;
3. *Flight Training Basic, Aviation University of Air Force, Changchun* 130062, *China*;
4. *School of Computer Science and Information Technology, Northeast Normal University, Changchun* 130117, *China*)

Abstract: A novel particle swarm optimization (DPSO) algorithm for solving the flexible job-shop scheduling (FJSP) was proposed by introducing random vector to improve the updating equation of DPSO. The experiments show that the proposed algorithm is superior to the temporal decomposition method and the classic genetic method.

Key words: evolutionary algorithm; particle swarm optimization algorithm; scheduling problem

流水作业调度(scheduling)问题在生产生活中应用广泛. 但精确求解调度问题已经被证明是一个 NP 难问题. 为了适应大规模调度问题的求解, 研究者开始考虑智能优化算法[1-3]. 文献[4]通过对群体生物行为的模拟提出了粒子群优化算法. 该算法容易实现、参数少、具有较强的全局寻优能力. 但传统的粒子群算法只能应用于连续型问题, 而流水作业调度问题是一个离散型问题. 本文通过借鉴遗传算法中交叉和变异的思想, 改进了粒子群算法的变异机制, 使其适应于调度问题的求解. 在两个有代表性的标准测试问题上测试了该算法的求解性能, 并将实验结果与传统的时序分解算法和经典遗传算法相比较[5-7]. 结果表明, 本文提出的算法所求解的质量优于传统算法.

1 流水作业调度问题描述

在流水作业调度问题中, 有 m 台机器和 n 个作业, 每个作业 i 包括 m 个必须依次处理的工序 $(O_{i,1},O_{i,2},\cdots,O_{i,m})$, 其中 $O_{i,k}$ 表示作业 i 的第 k 个工序. 调度问题要求确定这 n 个作业的最优加工顺序, 使总的加工所需时间最少. 本文假设每个工序 $O_{i,j}$ 可由机器集合 M_{ij} 中任一台机器处理, 从而不但

收稿日期: 2013-10-11.

作者简介: 付志军(1984—), 男, 汉族, 硕士研究生, 从事智能算法的研究, E-mail: fuzj11@mails.jlu.edu.cn.

基金项目: 国家自然科学基金(批准号: 40904053).

使问题便于研究，且更能反应实际工业生产中的调度问题.

本文用两个向量 $\boldsymbol{O}$ 和 $\boldsymbol{M}$ 对调度问题进行编码. $\boldsymbol{O}$ 的长度等于问题中工序的数量，它的每个位置都是$\{1,2,\cdots,n\}$中的一个整数 i，表示第 i 个作业的一个工序. 如果第 i 个作业有 n_i 个工序，则 i 在 $\boldsymbol{O}$ 中出现 n_i 次，第 k 个 i 即表示第 i 个作业的第 k 个工序. $\boldsymbol{M}$ 的长度与 $\boldsymbol{O}$ 相同，它的每个位置是处理 $\boldsymbol{O}$ 中对应位置工序机器的标号[8-11].

用 $T^B_{i,j}$ 表示工序 $O_{i,j}$ 的开始时间，$T^E_{i,j}$ 表示工序 $O_{i,j}$ 的结束时间，则 $T^B_{i,j}$ 的计算方式如下：

$$T^B_{i,j}=\begin{cases}T^E_{PM(i,j)}, & j=1,\\ \max\{T^E_{i,j-1},T^E_{PM(i,j)}\}, & 2\leqslant j\leqslant n_i,\end{cases}$$

其中 $PM(i,j)$表示工序 $O_{i,j}$ 的前一个工序.

2 改进的 PSO 算法

文献[12]给出了一个适用于无等待流水车间调度问题的离散粒子群优化算法 DPSO，其粒子运动方程如下：

$$\boldsymbol{x}_i(t+1)=c_2\oplus F_2(c_1\oplus F_1(\omega\otimes F_0(\boldsymbol{x}_i(t)),\boldsymbol{p}_i(t)),\boldsymbol{g}_i(t)),$$

其中 $\lambda_i(t+1)=\omega\otimes F_0(\boldsymbol{x}_i(t))$表示粒子速度. 算法 DPSO 随机生成[0,1]间的实数，如果 $r\leqslant\omega$，则对 $\boldsymbol{x}_i(t)$进行变异操作得 $\lambda_i(t+1)=F_0(\boldsymbol{x}_i(t))$，否则 $\lambda_i(t+1)=\boldsymbol{x}_i(t)$. 这样粒子的各位置在相邻时刻要么同时变，要么同时不变，不能体现各位置的差异. 本文用速度向量 $\boldsymbol{v}_i(t)$代替实数型常量 ω，将粒子运动方程变为如下形式：

$$\begin{aligned}&\boldsymbol{v}_i(t)=\boldsymbol{x}_i(t)-\boldsymbol{x}_i(t-1),\\ &\boldsymbol{x}_i(t+1)=c_2\oplus F_2(c_1\oplus F_1(\boldsymbol{v}_i(t)\otimes F_0(\boldsymbol{x}_i(t)),\boldsymbol{p}_i(t)),\boldsymbol{g}_i(t)),\end{aligned}\tag{1}$$

其中：$\boldsymbol{v}_i(t)$表示粒子 i 在 t 时刻的速度，其长度与粒子编码长度相等；$\boldsymbol{x}_i(t)[j]-\boldsymbol{x}_i(t-1)[j]$取值为

$$\boldsymbol{x}_i(t)[j]-\boldsymbol{x}_i(t-1)[j]=\begin{cases}0, & \boldsymbol{x}_i(t)[j]\text{ 和 }\boldsymbol{x}_i(t-1)[j]\text{ 中所表示的工序和机器都相同},\\ 1, & \text{其他},\end{cases}$$

其他符号与文献[12]中运动方程的含义相同.

由于调度问题中对工序和机器均有约束，因此方程(1)中如果采用常规的交叉和变异操作，可能产生不符合要求的调度方案. 为了避免该问题，本文需要设计特殊的交叉和变异操作. 在交叉算法中，对两个父本 P_1 和 P_2，首先随机生成两个交叉位 s 和 t，$1\leqslant s<t\leqslant$ $\boldsymbol{O}$ 的长度. 对 P_1 中的工序 $O_{i,j}$，如果$O_{i,j}$在 P_2 中的位置位于 s 和 t 间，则在 P_1 中将处理 $O_{i,j}$的机器换为 P_2 中处理 $O_{i,j}$的机器，并保持 P_1 中其他工序对应的机器不变，然后将交叉后的 P_1 视为新的粒子.

算法 1 交叉算法. /* 对 P_1 和 P_2 执行交叉操作，结果赋给 P_1 */

1) 随机生成两个位置 s 和 t，记 P_2 中 s 和 t 间的工序集合为 O_{subst}，其对应的机器集合为 M_{subst}；

2) 用 M_{subst} 代替 P_1 中与 O_{subst} 中工序对应的工序处理机器，并保持 P_1 中其他工序对应的机器不变.

在变异算法中，首先随机生成两个位置 s 和 t，$1\leqslant s<t\leqslant\boldsymbol{O}$ 的长度. 设父本 P 中第 s 和 t 个位置对应的分别是作业 J_s 和 J_t 中的工序，变异算法将第 s 和 t 个位置对应的工序分别变为作业 J_t 和 J_s 中的工序，并保持作业 J_s 和 J_t 中各工序的相对顺序不变(不是简单地将这两个位置对应的工序进行对换，否则可能改变作业工序顺序，产生不符合要求的调度方案)，同时调整 $\boldsymbol{M}$ 中机器的位置，使各工序对应的机器保持不变. 算法 1 描述了交叉的过程，可用下例说明.

设 P_1，P_2 为两个调度方案，其工序向量和机器向量分别为

$$P_1:\ \boldsymbol{O}:\ 1\ \ 2\ \ 1\ \ 3\ \ 1\ \ 2\ \ 2\ \ 3,\qquad \boldsymbol{M}:\ 4\ \ 3\ \ 1\ \ 3\ \ 2\ \ 1\ \ 4\ \ 2;$$

$$P_2:\ \boldsymbol{O}:\ 3\ \ 2\ \ 2\ \ 1\ \ 3\ \ 1\ \ 2\ \ 1,\qquad \boldsymbol{M}:\ 2\ \ 4\ \ 1\ \ 2\ \ 3\ \ 1\ \ 3\ \ 4.$$

position_i position_j

通过随机选取交叉位置有

$$O_{\text{subst}}: O_{2,2}\ O_{1,1}\ O_{3,2},$$
$$M_{\text{subst}}: 1 \quad 2 \quad 3.$$

交叉算法就是在 P_1 的位于交叉位置间的工序做下述变换：

1）将处理工序 $O_{2,2}$ 的机器用机器 1 代替，则有

$$P_1: \boldsymbol{O}: 1\ \ 2\ \ 1\ \ 3\ \ 1\ \ 2\ \ 2\ \ 3,$$
$$\boldsymbol{M}: 4\ \ 3\ \ 1\ \ 3\ \ 2\ \ 1\ \ 4\ \ 2;$$

2）将处理工序 $O_{1,1}$ 的机器用机器 2 代替，则有

$$P_1: \boldsymbol{O}: 1\ \ 2\ \ 1\ \ 3\ \ 1\ \ 2\ \ 2\ \ 3,$$
$$\boldsymbol{M}: 2\ \ 3\ \ 1\ \ 3\ \ 2\ \ 1\ \ 4\ \ 2;$$

3）将处理工序 $O_{3,2}$ 的机器用机器 3 代替，则有

$$P_1: \boldsymbol{O}: 1\ \ 2\ \ 1\ \ 3\ \ 1\ \ 2\ \ 2\ \ 3,$$
$$\boldsymbol{M}: 2\ \ 3\ \ 1\ \ 3\ \ 2\ \ 1\ \ 4\ \ 3.$$

算法 2　变异算法. /* 变异后的工序向量为 $\boldsymbol{O}'$，机器向量为 $\boldsymbol{M}'$ */

1）随机生成两个位置 s 和 t，记 $\boldsymbol{O}[s]=J_s$，$\boldsymbol{O}[t]=J_t$，处理作业 J_s 所有工序的机器序列为 MO_s，处理作业 J_t 所有工序的机器序列为 MO_t；

2）计算变异后的工序向量：

$$\boldsymbol{O}'[i]=\begin{cases} J_t, & i=s, \\ J_s, & i=t, \\ \boldsymbol{O}'[i]=\boldsymbol{O}[i], & \text{其他}; \end{cases}$$

3）计算变异后的机器分配向量 $\boldsymbol{M}'$.

① 对所有的 $\boldsymbol{O}'[i]\neq J_s$ 且 $\boldsymbol{O}'[i]\neq J_t$，置 $\boldsymbol{M}'[i]=\boldsymbol{M}[i]$；

② 对所有的 $\boldsymbol{O}'[i]=J_i$，将机器序列 MO_i 中的机器编号按从前向后顺序依次填入对应的 $\boldsymbol{M}'[i]$中；

③ 对所有的 $\boldsymbol{O}'[i]=J_j$，将机器序列 MO_j 中的机器编号按从前向后顺序依次填入对应的 $\boldsymbol{M}'[i]$中.

算法 2 描述了变异的过程，可用下例说明.

设 P 为一个调度方案，其工序向量和相应的机器向量如下：

$$\boldsymbol{O}: 2\ \ 1\ \ 1\ \ 3\ \ 1\ \ 2\ \ 2\ \ 3,$$
$$\boldsymbol{M}: 1\ \ 4\ \ 3\ \ 3\ \ 2\ \ 1\ \ 2\ \ 4.$$

随机选取两个变异位置，首先计算变异后的工序向量：

$$\boldsymbol{O}': 2\ \ 1\ \ 3\ \ 3\ \ 1\ \ 2\ \ 2\ \ 1;$$

然后计算变异后的机器分配向量 $\boldsymbol{M}'$，从而得到新的调度方案：

$$\boldsymbol{O}': 2\ \ 1\ \ 3\ \ 3\ \ 1\ \ 2\ \ 2\ \ 1,$$
$$\boldsymbol{M}': 1\ \ 4\ \ 3\ \ 4\ \ 3\ \ 1\ \ 2\ \ 2.$$

3　实验结果

采用文献[7]中 8×8 的部分 FJSP 与 10×10 的完全 FJSP 对所提出的离散粒子群算法 MDPSO 进行测试. 实验参数设置为 $P_{\text{size}}=80$，$c_1=c_2=0.5$，算法最大迭代次数为 300，优化目标是使调度方案的完工时间(makespan)最短. 将实验结果与时序分解方法[5]和经典的遗传算法[6]相比较，各算法得到调度方案的完工时间列于表 1.

表 1　不同算法的运行时间比较(工时)

Table 1　Comparison of the running time (man-hour) with different algorithms

问题规模	时序分解算法	经典遗传算法	MDPSO 算法
8×8	19	16	14
10×10	16	7	7

对8×8的部分FJSP，MDPSO求得的调度方案只需要14工时，优于时序分解算法的19工时和经典遗传算法的16工时；对10×10的完全FJSP，MDPSO求得的调度方案只需要7工时，与经典算法相同，但远优于时序分解算法的16工时.

参 考 文 献

[1] 关凇元，刘大有，金弟，等. 基于局部搜索的遗传算法求解自动组卷问题[J]. 吉林大学学报：理学版，2009，47(5)：961-968.（GUAN Songyuan，LIU Dayou，JIN Di，et al. Genetic Algorithm with Local Search for Automatic Test Paper Generation[J]. Journal of Jilin University：Science Edition，2009，47(5)：961-968.）

[2] 周屹，李海龙，王锐. 遗传算法求解物流配送中带时间窗的VRP问题[J]. 吉林大学学报：理学版，2008，46(2)：300-303.（ZHOU Yi，LI Hailong，WANG Rui. VRP Problem with Time Windows in the Logistics and Distribution Solved by Genetic Algorithm[J]. Journal of Jilin University：Science Edition，2008，46(2)：300-303.）

[3] Ercan M F，LI Xiang. Particle Swarm Optimization and Its Hybrids[J]. International Journal of Computer and Communication Engineering，2013，2(1)：52-55.

[4] Kennedy J，Eberhart R C. Particle Swarm Optimization[C]//Proceedings of the IEEE International Conference on Neural Networks Ⅳ. Piscataway：IEEE Press，1995：1942-1948.

[5] Angeline P J. Evolutionary Optimization versus Particle Swarm Optimization：Philosophy and Performance Difference[C]//Proc of the 7th Annual Conf on Evolutionary Programming. Berlin：Springer，1998：601-610.

[6] Yoshida H，Kawata K，Fukuyama Y，et al. A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment[J]. IEEE Trans on Power Systems，2000，15(4)：1232-1239.

[7] Kacem I，Hammadi S，Borne P. Approach by Localization and Multiobjective Evolutionaryoptimization for Flexible Job-Shop Scheduling Problems[J]. IEEE Transactions on Systems，Man，and Cybernetics，Part C：Applications and Reviews，2002，32(1)：1-13.

[8] Aarts E H L，Laarhoven P J M，Van，Lenstra J K，et al. A Computational Study of Local Search Algorithms for Job Shop Scheduling[J]. ORSA J on Comput，1994，6(2)：118-125.

[9] Nowicki E，Smutnicki C. An Advanced Tabu Search Algorithm for the Job Shop Problem[J]. Journal of Scheduling，2005，8(2)：145-159.

[10] Danna E，Rothberg E，Pape C L. Exploring Relaxation Induced Neighborhoods to Improve MIP Solutions[J]. Mathematical Programming，2005，102(1)：71-90.

[11] Lourenco H R. Job-Shop Scheduling：Computational Study of Local Search and Large-Step Optimization Methods[J]. European Journal of Operational Research，1995，83(2)：347-367.

[12] PAN Quanke，Tasgetiren M F，LIANG Yunchia. A Discrete Particle Swarm Optimization Algorithm for the No-Wait Flowshop Scheduling Problem[J]. Computers & Operations Research，2008，35(9)：2807-2839.

（责任编辑：韩 啸）