doi:10.3772/j.issn.1002-0470.2014.08.014

# 基于 MPSO 的有限缓冲区多产品厂间歇调度问题的研究①

李青青② 徐震浩③ 顾幸生

(华东理工大学化工过程先进控制和优化技术教育部重点实验室 上海 200237)

**摘 要** 研究了以最小化最大完工时间为目标的有限缓冲区多产品厂间歇调度问题,提出了一种基于多种群粒子群优化(MPSO)的间歇调度算法。该算法采用多种群,增加了种群初始粒子的多样性,在每一代子种群并行进化的过程中引入移民粒子,使子种群之间相互影响和促进,避免算法过早地陷入局部最优,提高了算法的全局搜索能力;每代进化后选出子种群中的优秀粒子作为精华种群,并对其进行变邻域搜索(VNS),进一步提高了算法的收敛精度。通过对不同规模调度问题的仿真,以及与其它算法的对比,证明了该算法解决有限缓冲区多产品厂间歇调度问题的有效性和优越性。

**关键词** 多种群粒子群优化(MPSO),有限缓冲区,间歇调度,移民粒子,变邻域搜索(VNS)

## 0 引 言

间歇调度是指间歇生产过程的调度,是在一定的设备(反应器、分离装置、中间储罐等)和其他条件的限制下,确定各产品在不同设备上的加工次序和时间,使某个经济指标或性能指标达到最优。间歇生产过程也被称为批处理生产过程,典型的间歇生产过程分为多产品厂和多目的厂两种形式。在多产品厂中,生产方式比连续生产过程更加灵活,操作柔性更高,同一流程线可以生产各种规格不同的产品。因此多产品厂间歇生产过程特别适合于多品种、小批量、工艺复杂和附加值高的化学品[1]。本研究针对有限缓冲区多产品厂的间歇调度问题,以最小化产品的最大完工时间为优化目标,设计了一种基于多种群粒子群优化(mutil-swarm particle swarm optimization,MPSO)的间歇调度算法,并通过仿真实验验证了其有效性。

## 1 相关研究

在间歇生产过程中,不同性质的中间产品需要不同的中间存储策略,不同的中间存储策略会对调度结果产生不同的影响[2]。对于稳定的中间产品,在以往的研究中,很多都是假设工件可以在机器间的缓冲区内等待,也就是采用无限中间存储(unlimited intermediate storage,UIS)策略[3]。在实际生产过程中,要考虑到生产时间和生产成本等众多问题,可以采用有限中间存储(finite intermediate storage,FIS)策略,限制缓冲区的大小,能够比较折中地处理这些问题。鉴于有限缓冲区间歇调度问题在学术与工程应用领域的重要性,开发高效和新颖的有限缓冲区间歇调度技术很值得关注[4]。

目前,对该类问题已经有了很多研究成果。为解决短期间歇调度问题,Zhu 等人[5]提出了一种新型粒子群优化(particle swarm optimization,PSO)算法,该 PSO 算法引入了变异算子、交叉操作等提高算法的局部搜索能力。Liu 等人[6]用混合 PSO 算法解决含有限缓冲区的调度问题,引入基于升序排列(ranked-order-value,ROV)规则,并采用基于问题的有向图模型和基于自适应学习策略的模拟退火两种局部搜索方式,丰富了算法的搜索行为,增强了算法的局部开挖能力。Lei 等人[7]采用二进制锦标赛选择和外部存档更新策略的遗传算法(genetic algorithm,GA),对间歇调度问题进行求解。Duan 等人[8]使用离散微分进化算法解决有限缓冲区调度

① 国家自然科学基金(61104178,61174040)资助项目。

② 女,1989 年生,硕士生;研究方向:生产调度,智能计算;E-mail:liqing0221@163.com

③ 通讯作者,E-mail:xuzhenhao@ecust.edu.cn

(收稿日期:2014-05-06)

问题,应用启发式(nawaz-enscore-ham,NEH)算法产生初始种群,在迭代的过程中采用变异和交叉操作产生新的候选解。胡蓉等人[9]提出了一种混合差分进化算法,把最优计算量分配和假设检验与差分进化算法相结合,使算法有较好的鲁棒性,但是算法的收敛速度较慢。Zeng 等人[10]设计了一种自适应细胞自动变异粒子群优化算法解决生产线间歇调度问题,该算法能够有效地避免优化结果过早地陷入局部最优解。Tang 等人[11]在建立了混合整数线性模型的基础上提出了一种分散搜索算法来处理阻塞间歇调度问题,该算法将参考集分为三部分增加解的多样性,使用多种搜索策略,提高了算法的局部搜索能力。

本文提出用一种改进的多种群粒子群优化算法来求解采用 FIS 方式生产的多产品厂间歇调度问题。以往的多种群粒子群优化算法,每个种群之间并没有什么联系,作为独立的单元进行进化[12,13],本文提出的改进多种群粒子群优化算法在种群之间增加了移民粒子,各个子种群之间相互影响促进,增加了种群多样性,能够有效地避免优化结果过早地陷入局部最优,同时对每代选出的精华种群进行局部搜索,提高了算法的收敛精度。

## 2 问题描述

多产品厂间歇生产过程调度一般做如下假设:(1)产品的批次是固定的,多种产品遵循同样的加工路径;(2)在加工过程中每批产品不能和其它批次的产品混合,也不能分开在不同的设备上处理;(3)原料和产品有足够的存储容量。每批产品在每个加工单元的加工时间包括了进出料时间和清洗时间,优化目标为最小化总流程时间。

在以上假设的基础上,有限缓冲区多产品厂间歇调度过程可以描述为:有 $n$ 个待加工产品依次在 $m$ 个设备单元上加工,一个产品在某一时刻只能在一个设备上加工,一个设备在某一时刻只能加工一个产品,在每台设备上, $n$ 个产品的加工次序都相同;每两台相邻设备 $j$ 和 $j-1$ 之间的缓冲区大小为 $B_j$, 当产品 $i$ 在设备 $j-1$ 上完成后,如果下一台设备 $j$ 正在使用,则产品 $i$ 进入缓冲区,如果此时缓冲区已满,则产品 $i$ 被阻塞到当前机器上,产品在缓冲区中服从先进先出的规则。

已知产品 $i$ 在设备 $j$ 上的加工时间为 $t_{ij}$,$S_{ij}$ 表示第 $i$ 个产品在第 $j$ 个设备单元上的开始加工时间,其中 $i \in \{1,2,\cdots,n\}$, $j \in \{1,2,\cdots,m\}$, 假设 $\prod = (k_1,k_2,\cdots,k_n)$($k_i$ 为整数,且 $1 \leqslant k_i \leqslant n$)为所有产品的一个排序, $C(k_i, j)$ 表示产品 $k_i$ 在设备 $j$ 上的完工时间,则 $C(k_n,m)$ 表示此工序的最大完工时间,有限缓冲区多产品厂间歇调度的数学模型为 $\min(C(k_n,m))$。$C(k_n,m)$ 的递推公式如下:

$$S_{k_1,1} = 0 \tag{1}$$

$$S_{k_1,j} = S_{k_1,j-1} + t_{k_1,j-1},\ j = 2,\cdots,m \tag{2}$$

$$S_{k_i,1} = S_{k_{i-1},1} + t_{k_{i-1},1},\ i = 2,\cdots,B_{j+1}+1 \tag{3}$$

$$S_{k_i,j} = \max\{S_{k_i,j-1} + t_{k_i,j-1}, S_{k_{i-1},j} + t_{k_{i-1},j}\},\quad i = 2,\cdots,B_{j+1}+1,\ j = 2,\cdots,m \tag{4}$$

$$S_{k_i,1} = \max\{S_{k_{i-1},1} + t_{k_{i-1},1}, S_{i-B_2-1,2}\},\quad i > B_{j+1}+1 \tag{5}$$

$$S_{k_i,j} = \max\{S_{k_i,j-1} + t_{k_i,j-1}, S_{k_{i-1},j} + t_{k_{i-1},j}, S_{i-B_{j+1}-1,j+1}\},\quad i > B_{j+1}+1,\ j = 2,\cdots,m \tag{6}$$

$$C(k_i,j) = S_{k_i,j} + t_{k_i,j},\quad i = 1,\cdots,n,\ j = 1,\cdots,m \tag{7}$$

在上述公式中,式(1)(2)表示产品 $k_1$ 在每台设备上的开始加工时间;式(3)(4)表示产品 $k_i$ 在设备 $j = 1,2,\cdots,m(i = 2,3,\cdots,B_{j+1}+1)$ 上的开始加工时间,保证了产品在设备上加工完成后不占有设备;式(5)和式(6)表示产品 $k_i$ 在设备 $j = 1,2,\cdots,m(i > B_{j+1}+1)$ 上的开始加工时间,此时需要考虑缓冲区的大小;式(7)表示每个产品在每台设备上的完成时间。有限缓冲区多产品厂间歇调度的最终目标是找到最优序列使得 $C(k_n,m)$ 最小。由上述公式可知,如果缓冲区的大小满足 $B_j \geqslant n-1$, 则相当于缓冲区无限大,对最大完工时间指标没有任何影响。

## 3 多种群粒子群优化(MPSO)算法

从 1995 年 Kennedy 和 Eberhart[14]提出粒子群优化(PSO)算法以来,PSO 受到了广泛关注。该算法概念简单、实现方便,但也存在一些问题,主要是在复杂问题的优化中,易于早熟。因此,许多学者提出了不少改进的粒子群算法。特别是美国的 Shi 和 Eberhart[15]提出了带有惯性权重的改进粒子群算法。在迭代的过程中,惯性权因子 $w$ 在一定范围内线性变化,这一改进的粒子群算法一般作为基本粒子群优化(general particle swarm optimization,GPSO)算法。GPSO 算法按照如下公式进行各粒子速度和

位置的更新：

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_1 (p_{ij}^{k} - x_{ij}^{k}) + c_2 r_2 (p_g^{k} - x_{ij}^{k}) \quad (8)$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1},\ j = 1,2,\cdots,m \quad (9)$$

GPSO 算法在搜索时，所有粒子总是跟踪当前个体最优值 $p_{ij}$ 和全局最优值 $p_g$，大多数粒子都聚集在 $p_g$ 附近，使群体多样性降低，容易陷入局部最优。本文针对 GPSO 算法存在的问题，提出了一种多种群粒子群优化(MPSO)算法。

### 3.1 算法思路

突破 GPSO 算法仅靠单个群体进行进化操作，MPSO 算法引入多个子种群同时进行优化搜索，每个子种群的进化方式使用 GPSO 算法，但是跟踪各自的群体最优，保证了群体的多样性；选出各个种群的优秀粒子作为移民粒子，将目标种群中较差的个体用移民粒子替换，各个子种群之间通过移民粒子进行联系，实现多种群的协同进化，避免进化结果过早地陷入局部最优；每进化一代，从各个子种群中选出最优粒子放入精华种群，然后对精华种群进行变邻域搜索(variable neighborhood search, VNS)，再将优化后的最优粒子返回到各个子种群，使整体能够更快地收敛到全局最优。MPSO 算法每代子种群粒子进化机制如图 1 所示。

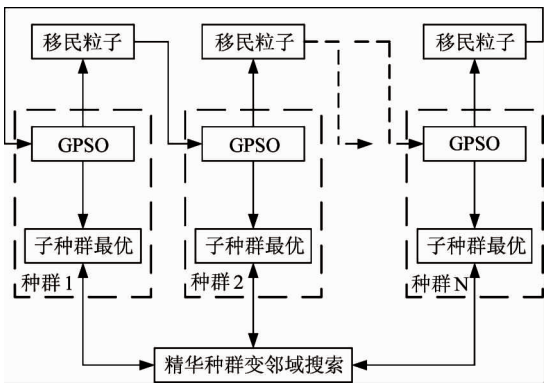


**图 1 MPSO 每代子种群粒子进化结构示意图**

如图中所示，每代 1 ~ $N$ 个子种群粒子更新机制都采用 GPSO，惯性权因子 $w$ 按相同的线性规律变化，加速常数 $c_1$ 和 $c_2$ 取相同的值。每代子种群粒子按照上图更新完成后，若没有满足终止条件，各子种群进入下一代的进化。

本文提出的多种群粒子群优化算法，并没有改变粒子的进化方式，保持了粒子群优化算法的简单易操作性，同时引入移民粒子和精华种群，避免粒子过早地陷入局部最优，提高群体的全局搜索能力。

### 3.2 算法流程

MPSO 算法采用随机方式生成初始种群，该方法能够覆盖较大的解空间，同时，将初始解分为多个种群分别进化，能够保证种群粒子的多样性，有利于全局优化。

MPSO 算法的流程如下：

(1)首先设定 MPSO 算法的参数，包括子种群个数 $N$、子种群内粒子数 $NIND$、移民粒子数 $M$、最大进化代数 $nMAX$，以及 $c_1$、$c_2$、$w(k)$、$x$ 和 $v$ 的取值范围。

(2)对每个子种群粒子的 $x$ 和 $v$ 进行随机初始化，并计算各初始种群个体的适应度值，保存当前每个粒子的位置为个体极值，将每个子种群中适应度最优的粒子保存为每个子种群的种群极值。

(3)各个子种群按照图 1 所示进化机制进行搜索，并更新子种群的种群最优值、粒子个体最优值和全局最优值。

(4)以达到最大进化代数为终止条件，如不满足，返回(3)进行下一代子种群进化。

(5)记录最后一代精华种群中适应度值最优个体的位置和适应度值，即为全局最优值。

每一代种群进化过程中，最重要的是移民粒子的加入和精华种群的进化和回归，使各个子种群之间有了信息的交流，来避免优化结果过早地陷入局部最优。

根据上文的介绍，MPSO 的每一代种群进化过程如下：

(1)各个子种群个体按公式(8)、(9)进行位置和速度更新。

(2)选择每个种群的移民粒子，进行移民。

(3)选择各个种群的最优粒子放入精华种群，并对精华种群的粒子进行变邻域搜索。

(4)将精华种群粒子分别返回到各个子种群，作为子种群的局部最优值，同时更新粒子个体最优值。

(5)没有达到最大进化代数，返回(1)继续进化。

## 4 仿真实验

### 4.1 实验设置

为了调试算法的参数和检验算法的性能，在双核 Intel Core 2.20GHz、内存为 1.99GB 的 PC 机上进行实验。采用 Matlab R2008a 对 MPSO、GPSO 和 GA

算法编程,对 26 个不同规模的典型调度问题求解,并根据规模的大小来确定合适的进化代数。每种情况运行 10 次,通过结果分析和对比,选出对有限缓冲区多产品厂调度问题具有普遍适用性的 MPSO 参数。

### 4.2 缓冲区大小对生产调度的影响

在缓冲区有限的情况下,当一件产品完成了在某个加工单元的加工后,若该单元下游的缓冲区已满,这个产品就被阻塞了,就会滞留在这个加工单元,所以缓冲区的大小会对生产产生一定的影响。当 $B = 0$ 时,调度问题可以认为是一个阻塞调度问题,随着 $B$ 的不断增加,其对完工时间的影响也会越来越小,当 $B > 4$ 时,最大完工时间的值和 $B$ 为无穷时差别不大[16]。所以本文主要考虑 $B = 1$ 和 $B = 4$ 时情况,然后和 $B = 0$ 的情况进行对比,每种情况运行 10 次,求其平均值和最小值,结果如表 1 所示。

**表 1 缓冲区大小不同时仿真结果对比**

| $n \times m$ | $B = 0$ | | $B = 1$ | | $B = 4$ | |
|---|---|---|---|---|---|---|
| | avg | min | avg | min | avg | min |
| 12 ×5 | 906 | 903 | **879** | **879** | **879** | **879** |
| 14 ×4 | 1076.7 | 1063 | 971.2 | **964** | **969.6** | 969 |
| 10 ×6 | 962.2 | 961 | 936.3 | **933** | **935.7** | **933** |
| 8 ×9 | 1021 | 1021 | **1017** | **1017** | **1017** | **1017** |
| 20 ×15 | 2140.5 | 2120 | 2010.3 | 1985 | **2001.2** | **1980** |
| 30 ×10 | 2602.2 | 2571 | 2300.9 | 2272 | **2284.7** | **2251** |
| 20 ×5 | 1390.6 | 1376 | 1301.8 | **1301** | **1301** | **1301** |
| 20 ×10 | 1763.9 | 1733 | 1662.9 | 1637 | **1660.8** | **1636** |
| 20 ×20 | 2315.4 | 2286 | 2188.8 | 2165 | **2178.6** | **2155** |
| 50 ×5 | 3207.6 | 3187 | 2892.6 | **2892** | **2892** | **2892** |
| 50 ×10 | 3548.2 | 3528 | 3050.7 | 3015 | **3002.1** | **2976** |
| 50 ×20 | 4431.4 | 4386 | 3972.8 | 3943 | **3932.7** | **3899** |
| 75 ×20 | 6136.6 | 6078 | 5293.8 | 5244 | **5206.5** | **5148** |
| 100 ×30 | 8785 | 8755 | 7661.1 | 7619 | **7497.3** | **7460** |

由表 1 数据可以看出,加工规模越大,有限缓冲区在调度中的优势越明显。当 $B$ 由 0 增加到 1 的时,优化结果减小的幅度特别大,但当 $B$ 从 1 进一步增加到 4 时,优化结果只有小幅度的减小。因此,在实践中,要兼顾生产效率和生产成本,可以通过适当增加缓冲区来实现。下文的研究都设定 $B = 1$,通过以下实验对改进算法进行参数选择。

### 4.3 参数选择

对于启发式算法来说,参数的选择会对算法的性能产生极大的影响,一般通过仿真实验来确定较优的参数组合。因此本文通过对不同参数组合的测试,来确定 MPSO 算法的最佳参数。MPSO 算法有 3 个参数:子种群的个数 $N$,子种群内粒子的个数 $NIND$,移民粒子的个数 $M$。测试问题选择 TA 类的 18 个实例,缓冲区 $B = 1$,用相对偏离(relative percentage deviation, RPD)误差作为评价指标。RPD 的计算公式如下

$$RPD = \frac{Obt_{sol} - Best_{sol}}{Best_{sol}} \times 100 \qquad (10)$$

其中,$Obt_{sol}$ 表示某个参数组合得到的最优解,$Best_{sol}$ 表示所有参数组合得到的最优解,某个参数组合得到的 RPD 的值越小,说明该参数组合的性能越好。首先令 $N = 10$,$NIND = 20$,移民粒子的个数 $M$ 分别取 0,1,2,3 和 4,每种情况重复 10 次试验,取 RPD 的平均值,试验结果如表 2 所示。表中将某个问题的最小 RPD 值加粗,Count 值表示在 $M$ 不同的情况下,所取得最小 RPD 值的个数。

由表 2 可以看出,当 $M = 2$ 时,得到了 13 次最小 RPD 值,并且可以发现,并非 $M$ 取值越大,所得到的解越好。显然,当 $M$ 为 0 时,各个子种群之间没有交流,虽然保证了粒子的多样性,但并不能充分体现粒子群算法的社会性,各个子种群只在自己的局部搜索最优值,很容易陷入局部最优,且收敛精度降低;当 $M = 2$ 时,各个子种群之间做了适当的交流,相互之间的信息交换,可以彼此影响和促进;当 $M$ 逐渐增大时,子种群之间的差异性减少,降低了种群多样性,容易使结果快速陷入局部最优,同时也会使搜索时间变长。表 2 充分说明了移民粒子存在的优

越性,因此,通过实验,可以确定当 $M$ 取值为 2 时,算法的性能达到最优。

由于在迭代代数相同的情况下,种群规模越大,计算时间越长,而参数 $NIND$ 和 $N$ 直接影响了种群规模的大小,所以在对这两个参数进行调试时,将迭代终止条件改为固定计算运行时间:$n \times m \times 0.2$s,$n$ 为相应问题的产品数,$m$ 为加工单元的数量。

**表 2 移民粒子个数 $M$ 不同取值下的 RPD 值**

| Problem | $n \times m$ | $M = 0$ | $M = 1$ | $M = 2$ | $M = 3$ | $M = 4$ |
|---|---|---|---|---|---|---|
| Ta001 | 20 × 5 | 0.0428 | 0.1427 | **0.0143** | 0.0428 | 0.2282 |
| Ta005 | 20 × 5 | 0.7872 | 1.0824 | **0.7257** | 0.7380 | 0.9225 |
| Ta009 | 20 × 5 | **0** | 0.0307 | **0** | **0** | 0.0307 |
| Ta011 | 20 × 10 | 1.1985 | 1.3039 | **0.8341** | 1.3998 | 1.3135 |
| Ta015 | 20 × 10 | 2.2599 | **1.9690** | 2.0757 | 2.1435 | 2.2502 |
| Ta019 | 20 × 10 | 1.8957 | 2.1534 | **1.5706** | 1.9816 | 2.2025 |
| Ta021 | 20 × 20 | 1.4908 | 1.5897 | **1.2929** | 1.4644 | 1.6359 |
| Ta025 | 20 × 20 | 1.6671 | 1.8355 | **1.5789** | 1.6775 | 1.6474 |
| Ta029 | 20 × 20 | 1.9123 | 1.9170 | **1.1718** | 2.0056 | 2.2108 |
| Ta031 | 50 × 5 | 1.1876 | 1.1570 | **0.9635** | 1.1009 | 1.1366 |
| Ta035 | 50 × 5 | 1.8315 | **1.3448** | 1.4390 | 1.5175 | 1.4547 |
| Ta039 | 50 × 5 | 0.1902 | 0.0761 | **0.0657** | 0.0761 | 0.1798 |
| Ta041 | 50 × 10 | 1.9459 | 1.8339 | 1.7266 | **1.7219** | 1.8805 |
| Ta045 | 50 × 10 | 2.4521 | 2.0177 | **1.6768** | 2.0037 | 1.8963 |
| Ta049 | 50 × 10 | 3.7512 | 2.8247 | **2.1575** | 3.1826 | 3.5544 |
| Ta051 | 50 × 20 | 1.6869 | 1.3297 | 1.2634 | 1.2707 | **1.2302** |
| Ta055 | 50 × 20 | 1.7232 | 1.4733 | **1.2868** | 1.3342 | 1.4196 |
| Ta059 | 50 × 20 | 1.9031 | 1.8396 | 1.8447 | **1.0962** | 1.7102 |
| Count | | 1 | 2 | 13 | 3 | 1 |

取定 $N = 10$,$M = 2$,$NIND$ 分别取 10、20、30、45、60,每种情况运行 10 次,统计每种情况下 RPD 取到最小值的个数(count),得到图 2。$NIND$ 的大小影响种群规模,进而影响到进化时间,从图中可以看出,$NIND$ 由 20 到 30 时,种群规模增大,粒子每代进化时间大大增加,使得进化代数减少,Count 值明显减小;随着 $NIND$ 值增大,种群规模持续增大,种群多样性也大大增加,优化结果有所好转但并不显著。所以,子种群粒子数 $NIND$ 选为 20 比较合适。

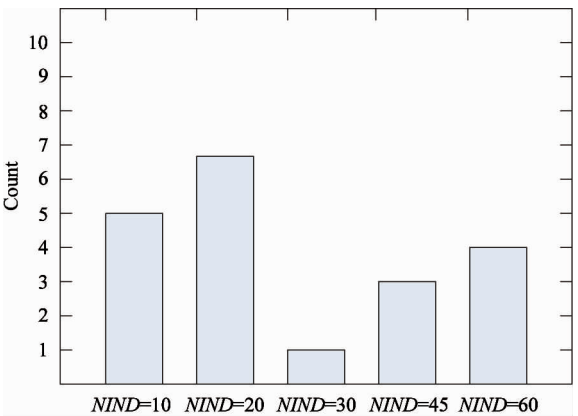


**图 2 不同子种群粒子数的 Count 值直方图**

取定 $NIND = 20$,$M = 2$,$N$ 分别取 5、10、15、20、25,每种情况运行 10 次,统计每种情况下 RPD 取到最小值的个数,得到结果如图 3 所示。从图中我们可以看到,$N$ 值太小,会影响粒子多样性;$N$ 值太大,会使每次进化时间增长,减少总进化代数,对寻找最优值产生很大的影响。当 $N = 15$ 时,得到的结果最好,也就是算法的性能最佳。

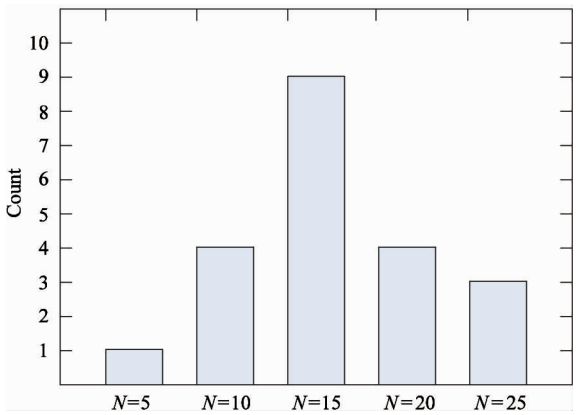


**图 3 不同子种群个数的 Count 值直方图**

经过以上数据实验,可以确定 MPSO 算法的三个参数的最终取值:子种群个数 $N = 15$,子种群内

粒子数 $NIND = 20$, 移民粒子的个数 $M = 2$。

### 4.4 算法比较与分析

通过上面的仿真实验,确定了 MPSO 算法的参数,为了进一步验证 MPSO 的有效性和优越性,测试中选择基本粒子群优化(GPSO)算法和遗传算法(GA)来与多种群粒子群优化(MPSO)算法进行比较。分别用这三种算法对不同规模的有限缓冲区多产品厂间歇调度问题进行求解,三种算法采用相同的粒子规模,相同的迭代次数。仿真的过程中,评价指标除了采用上文中的相对偏离(RPD)误差,还有最优相对误差(optimal relative error, ORE)和最差相对误差(worst relative error, WRE):

$$ORE = \frac{Obt_{opt} - Best_{sol}}{Best_{sol}} \times 100 \tag{11}$$

$$WRE = \frac{Obt_{wor} - Best_{sol}}{Best_{sol}} \times 100 \tag{12}$$

仿真过程中,每种情况运行 10 次, $Obt_{opt}$ 为 10 次中算法最优值, $Obt_{wor}$ 为 10 次中算法最差值, $Best_{sol}$ 为所有参数组合得到的最优解。评价函数值越小,表示优化结果越好。

**表 3 不同算法仿真结果对比**

| Problem | $n \times m$ | Algorithm | RPD | ORE | WRE | *nMAX* |
|---|---|---|---|---|---|---|
| | | GPSO | 0.65 | 0 | 1.84 | |
| Ta005 | 20 ×5GA | 0.81 | 0 | 1.54 | 200 | |
| | | MPSO | **0.06** | **0** | **0.31** | |
| | | GPSO | 3.73 | 2.64 | 5.09 | |
| Ta015 | 20 × 10 | GA | 3.74 | 2.76 | 4.48 | 200 |
| | | MPSO | **2.10** | **0.37** | **3.50** | |
| | | GPSO | 5.45 | 3.50 | 7.23 | |
| Ta025 | 20 × 20 | GA | 5.26 | 3.50 | 6.20 | 300 |
| | | MPSO | **2.33** | **1.26** | **3.73** | |
| | | GPSO | 1.64 | 0.17 | 3.94 | |
| Ta035 | 50 × 5 | GA | 1.48 | 0.31 | 3.60 | 300 |
| | | MPSO | **0.10** | **0** | **0.62** | |
| | | GPSO | 6.85 | 5.23 | 8.14 | |
| Ta045 | 50 × 10 | GA | 7.53 | 5.70 | 9.18 | 500 |
| | | MPSO | **1.81** | **0.37** | **2.81** | |
| | | GPSO | 4.73 | 3.60 | 6.57 | |
| Ta055 | 50 × 20 | GA | 4.95 | 3.76 | 6.06 | 500 |
| | | MPSO | **0.63** | **0.25** | **1.12** | |
| | | GPSO | 1.50 | 0 | 5.35 | |
| Car3 | 12 × 5 | GA | 1.57 | 0.46 | 4.55 | 200 |
| | | MPSO | **0** | **0** | **0** | |
| | | GPSO | 2.86 | 1.66 | 3.84 | |
| Car4 | 14 × 4 | GA | 2.93 | 1.87 | 3.94 | 200 |
| | | MPSO | **1.20** | **0.52** | **2.39** | |
| | | GPSO | 1.00 | 0.54 | 2.79 | |
| Car5 | 10 × 6 | GA | 1.48 | 0.64 | 2.47 | 200 |
| | | MPSO | **0.45** | **0** | **0.54** | |
| | | GPSO | 0.26 | 0 | 1.57 | |
| Car6 | 8 × 9 | GA | 0.37 | 0 | 0.79 | 200 |
| | | MPSO | **0** | **0** | **0** | |
| | | GPSO | 5.67 | 3.46 | 7.74 | |
| Rec13 | 20 × 15 | GA | 6.52 | 3.77 | 8.19 | 400 |
| | | MPSO | **2.54** | **1.32** | **4.07** | |
| | | GPSO | 7.67 | 4.83 | 9.97 | |
| Rec19 | 30 × 10 | GA | 7.65 | 6.22 | 9.79 | 500 |
| | | MPSO | **2..87** | **2.01** | **3.71** | |

选择不同规模的问题进行仿真计算,用 MPSO 解决有限缓冲区间歇生产调度。缓冲区 $B = 1$,算法参数 $c_1 = c_2 = 1.49445$,微粒的速度区间[ -2, 2],位置采用实数编码,$w(k)$ 由0.9线性变化为 0.4,$N = 15$,$NIND = 20$,$M = 2$。遗传算法的参数交叉概率为 0.9,变异概率为 0.02,初始种群规模和 MPSO 的一样,都为 300。仿真对比结果如表 3 所示,$nMAX$ 表示迭代的代数,根据生产规模适当调整,黑色加粗代表最优指标。

由表 3 的结果可以看出,在解决有限缓冲区多产品厂间歇调度问题上,GPSO 算法和 GA 的三个评价函数值比较大,搜索精度不高,都容易陷入局部最优值。MPSO 算法的优化结果明显优于 GPSO 算法和 GA,特别是加工规模较大时,MPSO 的优越性更加突出。

为了更加清晰地显示 MPSO 算法在收敛性上的优越性,选择两个规模问题(Rec13:20 × 15,Rec19:30 × 10)进行仿真,并绘制出三种算法的进化曲线图(图4、图5)。由图4和图5可以看出,GPSO算法和 GA 的搜索精度都不高,且 GA 的收敛过程比较慢,而从 MPSO 算法的收敛曲线可以看出其在迭代的过程中,能够不断跳出局部极小值,收敛效果明显优于 GPSO 算法和 GA。

## 5 结 论

本文针对有限缓冲区多产品厂间歇调度问题,以最小化产品的最大完工时间为优化目标,设计了一种多种群粒子群算法。该算法通过引入移民粒子来增强子种群之间的交流和促进,提高粒子的全局

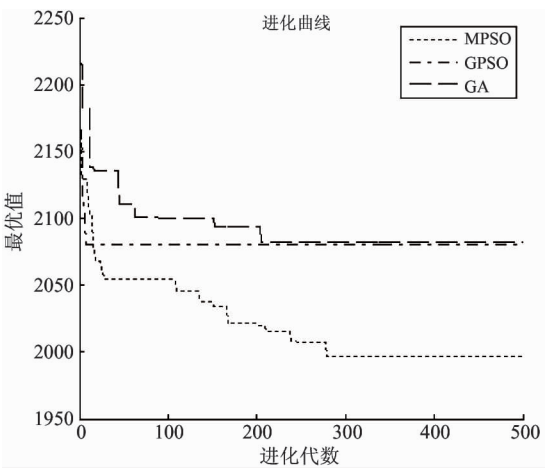


**图 4 Rec13:20 × 15 的收敛曲线**

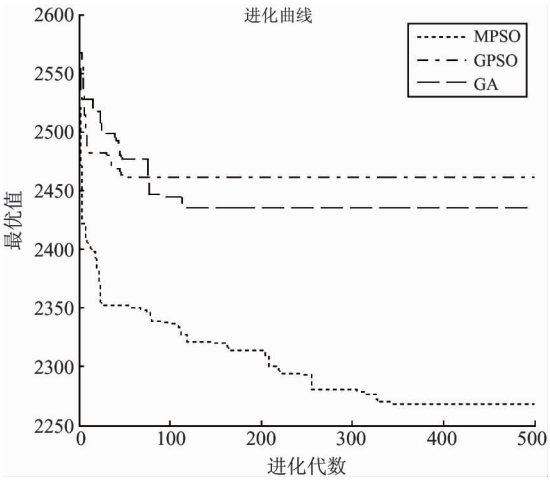


**图 5 Rec19:30 × 10 的收敛曲线**

搜索能力,并采用基于交换邻域的变邻域搜索算法对精华种群进行改良,使算法能够不断地跳出局部最优,寻找更优解。通过对不同规模的调度问题进行仿真实验,确定了算法中的参数,同时与标准粒子群算法和遗传算法的优化结果进行对比,验证了算法的有效性和优越性。本文对多产品厂间歇调度问题的研究进行了许多假设,但实际上,间歇调度问题存在许多不确定性,下一步将主要研究当一些条件不确定时,算法的有效性,同时对算法进行进一步的改进,以更好地解决多产品厂间歇调度问题。

**参考文献**

[1] 王万良,吴启迪. 生产调度智能算法及其应用. 北京:科学出版社, 2007. 192-197

[2] Liu S Q, Kozan E. Scheduling a flow shop with combined buffer conditions. *International Journal of Production Economics*, 2009, 117(2):371-380

[3] Hall N G, Sriskandarajah C. Survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 1996, 44(3):510-525

[4] Thornton H W, Hunsucker J L. A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage. *European Journal of Operational Research*, 2004, 152(1): 96-114

[5] Zhu J, Gu X S. A new particle swarm optimization algorithm for short-term scheduling of single-stage batch plants with parallel lines. In: Proceeding of the 6th International Conference on Intelligent Systems Design and Applications, Jinan, China, 2006.2:673-678

[6] Liu B, Wang L, Jin Y H. An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. *Computers and Operations Research*, 2008,35(9):2791-

2804

[7] Lei D M, Zhang Q F, Wen C, et al. Genetic algorithm based multi-objective scheduling in a flow shop with batch processing machines. In: Proceedings of the 2010 8th World Congress on Intelligent Control and Automation, Jinan, China, 2010. 694-699

[8] Duan J H, Qiao G Y, Zhang M. Solving the flow shop problem with limited buffers using differential evolution. In: Proceedings of the 2011 Chinese Control and Decision Conference, Mianyang, China, 2011. 1509-1513

[9] 胡蓉,钱斌. 一种求解随机有限缓冲区流水线调度的混合差分进化算法. 自动化学报, 2009, 35(12): 1580-1586

[10] Zeng M, Long Q Y, Liu Q M. Cellular automata variation particle swarm optimization algorithm for batch scheduling. In: Proceeding of the 2012 International Conference on Intelligent Systems Design and Engineering Applications, Sanya, China, 2012. 193-196

[11] Tang L X, Wang X P. A scatter search algorithm for a multistage production scheduling problem with blocking and semi-continuous batching machine. *Control System Technology*, 2011,19(5):976-989

[12] 吕林,罗绮,刘俊勇等. 一种基于多种群分层的粒子群优化算法. 四川大学学报,2008,40(5):172-174

[13] 李俊,孙辉,史小露等. 多种群粒子群算法与混合蛙跳算法融合的研究. 小型微型计算机系统,2013, 34(9): 2164-2168

[14] Kennedy J, Eberhart R C. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, 1995. 1942-1948

[15] Shi Y, Eberhart R C. A modified particle swarm optimizer. In: Proceedings of IEEE Congress on Evolutionary Computation, Piscataway, USA, 1998. 69-73

[16] Wang L, Zhang L, Zhang D Z. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers. *Computers and Operation Research*, 2006, 33(10): 2960-2971

# A Study of the MPSO-based batch scheduling with limited buffers

Li Qingqing, Xu Zhenhao, Gu Xingsheng

(Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, Shanghai 200237)

**Abstract**

For minimizing the total flow time of batch production, the bath scheduling problem with limited buffers was studied, and a batch scheduling algorithm based on the multi-swarm particle swarm optimization (MPSO) was proposed. The algorithm uses multiple swarms to increase the diversity of initial particles, and selects several good particles of each sub-swarm as the immigrant particles in the process of parallel evolution of sub-swarms to make the sub-swarms affect and promote each other, which prevents the result running into the local optimum prematurely and enhances the global research ability. It utilizes the variable neighborhood search (VNS) on the elite swarm consisting of each sub-swarm's best particle to further improve its convergence accuracy. The effectiveness of this algorithm was verified by the simulation of different scales of scheduling and the comparison of its performance with other algorithms. The proposed algorithm can solve the batch scheduling problem with limited buffers.

**Key words**: mutil-swarm particle swarm optimization (MPSO), limited buffers, batch scheduling, immigrant particle, variable neighborhood search (VNS)