差异工件平行机批调度问题的 SAGA*

舒 锋, 陈华平, 杜 冰, 李小林, 操张进
(中国科学技术大学 管理学院, 合肥 230026)

摘 要: 为了求解差异工件平行机批调度问题,提出了一种模拟退火遗传算法 (simulated annealing genetic algorithm,SAGA)。将模拟退火算法(simulated annealing,SA)的状态转移操作引入基于最优保留的遗传算法(genetic algorithm,GA)中,作为局部搜索算子,以避免算法陷入局部最优,也有效地发挥了 SA 和 GA 在局部搜索与全局搜索能力方面的优势。为了解决 GA 迭代后期适应函数难以区分一些适应度接近的个体这个问题,SAGA 分两阶段标定适应函数,在进化后期采用了一个加速适应函数。同时,将缺点较多的单切点交叉方式改换为效果更好的双切点交叉方式。实验结果表明,与以往文献中的 GA、BFLPT(best-fit longest processing time)和 FFLPT (first-fit longest processing time)启发式规则等相比, SAGA 是有效的。

关键词: 调度; 批处理机; 模拟退火遗传算法; 组合优化
中图分类号: TP301 **文献标志码**: A **文章编号**: 1001-3695(2010)12-4526-04
doi:10.3969/j.issn.1001-3695.2010.12.038

Minimizing makespan on parallel batching machines with non-identical job sizes using SAGA

SHU Feng, CHEN Hua-ping, DU Bing, LI Xiao-lin, CAO Zhang-jin
(*School of Management*, *University of Science & Technology of China*, *Hefei* 230026, *China*)

Abstract: This paper proposed a SAGA to minimize makespan on parallel batch processing machines with non-identical job sizes. Introduced the state transition operation in SA to improve local search in the GA based on an elitist selection strategy. By combining the advantages of both algorithms, SAGA was able to strike a balance between local search and global search, and wouldn't fall into local optimum. In addition, to solve the problem that GA was unable to effectively distinguish chromosomes with similar fitness values in the later iterations, this paper defined the fitness function in two stages. The fitness function in the second stage could accelerate the convergence of the algorithm. Further defined the crossover operator using more effective two-point crossover instead of single-point crossover. The experimental results show that the proposed SAGA outperforms FFLPT, BFLPT and traditional GA addressed in the previous work.

Key words: scheduling; batch processing machine; simulated annealing genetic algorithm; combinatorial optimization

0 引言

批调度问题是调度问题的一个重要分支。不同于经典调度问题,批调度问题中工件被分为很多个批,一个批中的工件可以在批处理机上同时处理,批的加工时间等于批中工件的最大加工时间,而经典调度问题中的机器一次仅仅处理一个工件。生产活动中有很多这样的例子,如金属加工业的热处理工序,半导体工业中的芯片预烧工序和汽车货运等。根据应用的不同,批调度问题又可以分为两种情况:a)工件的尺寸均为单位尺寸,机器的容量定义为它所能容纳的工件的个数;b)每个工件有其不同的容量需求,批中工件的总尺寸不能超过批的容量限制。通常把这两种情况分别称为同类工件的批调度问题和差异工件的批调度问题。目前,关于前者已有相当多的文献研究,而在差异工件批调度问题上的研究较少。

Uzsoy[1] 于 1994 年首先提出了差异工件单机批调度问题,并证明以最小化制造时间跨度 $C_{\max}$ 和最小化总完工时间 $\sum C_i$ 为目标的差异工件单机批调度问题是 NP-hard 的,具体的做法是:假设工件的加工时间是单位时间,简化问题模型,将问题规约为背包问题,而背包问题是 NP-hard 的。同时他还提出了几种启发式算法,其中 FFLPT 得到的结果较好。Dupont 等人[2] 提出了上述问题的一种分支定界算法,并给出了 BFLPT 和 SKP 两种新的启发式算法。Melouk 等人[3] 用模拟退火算法来解决该问题,并把结果与商业软件 CPLEX 的结果进行比较。Damodaran 等人[4] 采用顺序编码的编码方式,将遗传算法得到的较优的工件序列用 BF(best-fit)规则进行分批,结果优于模拟退火算法和 CPLEX。Kashan 等人[5] 提出了一种混合遗传算法,通过改进编码方式和采用 PSH 启发式规则对批中的工件进行交换来改进遗传算法得到的结果。

收稿日期: 2010-05-20; **修回日期**: 2010-06-28 **基金项目**: 国家自然科学基金资助项目(70821001);国家博士点基金资助项目(200803580024)

作者简介:舒锋(1987-),男,河南信阳人,硕士研究生,主要研究方向为智能优化与作业调度(shufeng@mail.ustc.edu.cn);陈华平(1965-),男,江苏无锡人,教授,博士,主要研究方向为智能算法、作业调度与云计算等;杜冰(1981-),男,安徽合肥人,博士研究生,主要研究方向为商务智能、智能算法;李小林(1985-),男,江苏邳州人,博士研究生,主要研究方向为智能算法;操张进(1986-),男,安徽安庆人,硕士研究生,主要研究方向为应急管理.

在现实生产活动中,通常是多台批处理机同时对批进行处理。Chang[6]、Kashan[7] 和 Damodaran 等人[8,9] 将问题从单机环境扩展到平行机环境下。批处理机是生产系统中的瓶颈[10],而且价格昂贵,所以必须采取有效的调度操作。在批调度问题中,制造时间跨度 $C_{\max}$ 经常被用来衡量调度的性能。本文主要研究了目标为最小化 $C_{\max}$ 的差异工件平行机批调度问题。

1　问题的数学模型

差异工件平行机批调度问题可以具体地作如下描述:

a) 待加工工件集合中有 n 个工件 $\{1,2,\cdots,n\}$,工件 i 的加工时间为 p_i,尺寸为 s_i。

b)有 m 台平行机,机器容量均为 C。在第 h 台机器上处理的批的集合记为 B_h。每一个批中工件的总尺寸不超过机器容量 C,假设没有尺寸大于机器容量的工件。

c) 批的加工过程不允许被中断,批 j 的加工时间 p_j 等于批中工件加工时间的最大值。

d) 第 h 台机器的制造时间跨度记为 $T_h(h=1,2,\cdots,m)$,$C_{\max}$ 等于 T_h 的最大值。

根据三参数表示法[11],目标函数是 $C_{\max}$ 的差异工件平行机批调度问题可以记为 $Pm|\text{batch},s_j\leqslant C|C_{\max}$。其数学模型为

$$\text{minimize } C_{\max}=\max_{1\leqslant h\leqslant m}\{T_h\} \tag{1}$$

subject to:

$$\sum_{j=1}^{k}x_{ij}=1,i=1,\cdots,n \tag{2}$$

$$\sum_{i=1}^{n}s_ix_{ij}\leqslant C,j=1,\cdots,k \tag{3}$$

$$p^j\geqslant p_ix_{ij},i=1,\cdots,n,j=1,\cdots,k \tag{4}$$

$$\sum_{h=1}^{m}y_{jh}=1,j=1,\cdots,k \tag{5}$$

$$T_h=\sum_{j\in B_h}p^j,h=1,\cdots,m \tag{6}$$

$$x_{ij}\in\{0,1\},i=1,\cdots,n,\ j=1,\cdots,k \tag{7}$$

$$y_{jh}=\{0,1\},j=1,\cdots,k,\ h=1,\cdots,m \tag{8}$$

$$\lceil\sum_{i=1}^{n}s_i/C\rceil\leqslant k\leqslant n,k\in Z^+ \tag{9}$$

式(2)限定了一个工件只能被分配到一个批里;式(3)保证了批中工件的总尺寸不能超过机器的容量限制;式(4)为批的加工时间约束;式(5)限定了一个批只可以分配到一个机器上;式(6)给出了每台机器的制造时间跨度;式(7)~(9)为决策变量,$x_{ij}=1$ 说明工件 i 被分配到批 j 中,否则 $x_{ij}=0$;$y_{jh}=1$ 说明批 j 被分配到机器 h 上,否则 $y_{jh}=0$;k 为总批数,$\lceil\sum_{i=1}^{n}s_i/C\rceil$ 表示不小于 $\sum_{i=1}^{n}s_i/C$ 的最小整数。

2　模拟退火算法与遗传算法

SA 是源于对热力学中退火过程的模拟,从一给定的初始较高温度开始,缓慢降低温度,在每个温度下,分子停留在状态 r_{old} 满足 Boltzmann 概率分布,由当前状态产生新状态 r_{new},一般采用 Metropolis 准则来决定是否接受新状态,多次重复这个状态转移操作,系统将趋于能量较低的平衡态,当温度趋近于零时,系统只能处于最小能量状态。在 SA 中,一个状态就是问题的一个解,系统相当于问题的解空间。通过缓慢降低温度参数,使 SA 能够在多项式时间内收敛到一个近似最优解。SA 局部搜索能力较强,且因为以一定概率接收较差解,所以能避免陷入局部最优。但由于对整个解空间不了解,SA 不能够很快落入最有希望的搜索区域,使得算法占用大量时间,运算效率较低。

GA 最早是由 Holland 提出来的,是一种应用非常广泛和成功的元启发式算法。一般认为,带有复制、交叉、变异操作的标准 GA 是不能收敛到全局最优解的[12],原因主要有两个:a)由于采用比例选择法,根据产生的随机数进行选择,可能会不正确反映个体适应度的选择而淘汰掉适应度较高的个体;b)当前种群中的最优个体可能在下一代种群中发生丢失,而且这种丢失现象会在进化过程中重复出现。Eiben 等人[13] 证明了保留最优个体的 GA 是概率性全局收敛的。GA 在迭代过程中同时保留了一组解而不是一个解,对解空间有很好的了解,因此其全局搜索能力较强。但是 GA 局部搜索能力较差,所以单纯地使用 GA 会耗费大量的时间。在进化后期,种群中个体适应度差别很小,算法的搜索能力变得非常差,而且 GA 容易陷入局部最优。

SA 与 GA 结合使用可以克服两个算法各自的缺点,发挥它们的优势。关于 SA 与 GA 结合的相关研究有很多[14,15]。在本文中,采取类似邢文训等人[16] 和王雪梅等人[17] 提出的 SA 与 GA 结合的思想,将 SA 算法的状态转移操作引入基于最优保留的 GA 中,作为 GA 的局部搜索算子,并改进 GA 的交叉方式和适应函数的标定方法,针对差异工件平行机批调度问题设计了一种模拟退火遗传算法,即 SAGA。

3　SAGA 及实现步骤

1)编码方式和初始种群的产生　SAGA 中采用顺序编码作为解的编码方式。使用 Damodaran 等人[4] 提出的方法生成算法的初始种群,即用 LPT(longest processing time)规则生成两个个体的染色体序列,再分别用 FF(first-fit)和 BF(best-fit)启发式规则分批,其余个体的染色体序列是随机生成的,并用 BF 启发式规则分批。其中 LPT 规则是将所有工件按照它们的加工时间降序排列。

FF 启发式规则可以描述为:依照顺序将工件放在现有第一个能够容纳该工件而不违反批处理机容量限制的批里。如果不存在这样的批,则新建一个批,并将该工件放入新建批中。重复该操作直到每个工件都被分配到一个批里。

BF 启发式规则非常类似于 FF 启发式规则,它们的区别在于 BF 启发式规则依照顺序把工件放入一个现有能容纳该工件而不违反机器容量限制的批里,且在所有能够容纳该工件而不违反机器容量限制的批里,该批中工件的总尺寸是最大的。

2)状态转移　对种群里的每一个个体状态 i,在它的邻域中随机选择一个新状态 $j\in N(i)$,按模拟退火中的接受概率:

$$A_{ij}=\min\{1,\exp(-\frac{f(j)-f(i)}{t_0})\} \tag{10}$$

接受或拒绝状态 j,若接受状态 j,则用 j 替换 i。其中:t_0 表示初始温度,A_{ij} 表示由状态 i 向 j 的转移概率,$f(i)$ 为状态 i 的目标值。选择邻域状态的方法是产生一个随机数 $e(2\leqslant e\leqslant n-1)$,调换个体编码中第 $e-1$ 位与 $e+1$ 位的值。

3)适应函数的标定　适应函数的标定分为两个阶段:

a)第一个适应函数。在算法迭代的前一阶段使用如下适应函数:

$$F_i = -f_i + f_{\max} + \varepsilon^k \tag{11}$$

其中：f_i 表示个体状态 i 目标值；$f_{\max}$ 表示种群中所有个体状态的最大目标值；ε 表示一个较小的数，目的是使种群中最差的个体仍有繁殖的机会，增加种群多样性；k 表示迭代次数。

b)第二个适应函数。算法迭代一定次数后，一些个体的目标值可能会很接近，上述适应函数已经很难区分这些个体，为了解决这个问题，在算法迭代的后期使用如下加速适应函数：

$$F_i = \exp\left\{-\frac{f_i - f_{\min}}{t_k}\right\} \tag{12}$$

其中：$f_{\min}$ 为种群中所有个体状态的最小目标值；t_k 为第 k 代的温度。

4)选择和交叉　采用遗传算法常用的轮盘赌来选择进行交叉的父代。Damodaran 等人[9]在文献中使用了单切点的交叉方式，单切点交叉方式总是仅仅交换染色体尾部的基因，信息量也比较小，交叉点位置的选择可能带来较大的偏差，为此本文改用双切点的交叉方式。图 1 和 2 分别为单切点与双切点交叉的一个实例，加粗的部分表示交叉点。

x	2	8	5	1	7	3	6	9	4	10
y	9	6	1	7	10	5	8	3	2	4

⇩

x	2	8	5	1	7	3	9	6	10	4
y	9	6	1	7	10	5	2	8	3	4

图1　单切点交叉的实例

x	2	8	5	1	7	3	6	9	4	10
y	9	6	1	7	10	5	8	3	2	4

⇩

x	2	8	6	1	7	5	3	9	4	10
y	6	9	1	7	10	5	8	2	3	4

图2　双切点交叉的实例

5)突变　对交叉后产生的每一个子代个体随机选择一个突变点，以概率 p_m 交换突变点左右的基因。图 3 为一个突变的实例。

x	2	8	6	1	7	5	3	9	4	10
y	6	9	1	7	10	5	8	2	3	4

⇩　突变点

x	2	8	6	1	5	7	3	9	4	10
y	6	9	1	7	5	10	8	2	3	4

图3　突变的一个实例

算法的实现步骤：

(a)产生初始种群，记录最优的个体；初始化参数 t_0，η，ε。

(b)迭代次数 $k=0$。

(c)对种群中每一个个体执行$\lceil n/10 \rceil$次状态转移操作，以式(10)计算接受概率，决定是否接受新状态。若产生了比记录中最优个体更优的解，则更新记录。

(d)计算个体 i 的选择概率

$$P_i = F_i / \sum_{i=1}^{M} F_i \tag{13}$$

其中：M 为种群规模。采用轮盘赌来选择进行双切点交叉的父代。共产生 M 个子代个体，以保持种群规模。每一个子代个体以概率 p_m 进行变异。$t_{k+1} = t_k \times \eta$。

(e)如果在新一代种群中有个体优于记录的最优个体，则更新记录。如果记录中最优个体优于新一代种群中最好的个体，则用记录的最优个体替换新一代种群中最差的个体。

(f)重复步骤(c)～(e)直至迭代次数 $k=N$。N 为最大迭代次数。

4　仿真实验

4.1　仿真实例

为了测试算法的有效性，需要考虑机器台数、工件规模、工件加工时间以及工件尺寸四个因素。本文采用类似于 Melouk 等人[3]提出的方法生成随机算例。各因素的记号、等级划分及取值范围如表 1 所示。实验总共设计了 48 类不同的子问题，每类子问题可用 $m_a J_b p_c s_d$ ($a=1,2$；$b=1,2,3,4$；$c=1,2$；$d=1,2,3$)表示。例如，$m_2 J_3 p_2 s_3$ 表示 4 台机器下，工件规模为 50，工件加工时间服从[1,20]离散均匀分布，工件尺寸服从[4,8]离散均匀分布的子问题。在所有子问题实例中，假设机器容量 C 都为 10。

表 1　问题因素及等级划分

因素	等级划分
机器台数	m_1 类：2 台机器；m_2 类：4 台机器
工件规模	J_1 类：10 个工件；J_2 类：20 个工件 J_3 类：50 个工件；J_4 类：100 个工件
工件加工时间	p_1 类：服从[1,10]离散均匀分布 p_2 类：服从[1,20]离散均匀分布
工件尺寸	s_1 类：服从[1,10]离散均匀分布(混合工件集) s_2 类：服从[2,4]离散均匀分布(小工件集) s_3 类：服从[4,8]离散均匀分布(大工件集)

实验中测试的算法包括 SAGA，Damodaran 等人[9]提出的 GA 以及两个性能较好的启发式算法，即 FFLPT 和 BFLPT。下界 LB 的计算方法[7]如下：记工件集 $J=\{q \mid C - s_q < \min_{i\in\{1,\cdots,n\}} s_i\}$，则 J 中的工件只能单独成批，然后，松弛剩余的工件，即加工时间为 p_i，尺寸为 s_i 的工件被松弛为 s_i 个加工时间为 p_i 的单位尺寸工件，对松弛后的工件按加工时间降序排列，依照顺序连续 C 个工件组成一个批，批数 $l=\lceil \sum_{i\notin J} s_i / C \rceil$，可得下界为

$$C_{\max}^{LB} = \max\left\{\left\lceil \sum_{i\in J} p_i + \sum_{j=1}^{l} p^j \right\rceil / m,\ \max_{i=\{1,\cdots,n\}}\{p_i\}\right\} \tag{14}$$

各算法采用 LBPT(longest batch processing time)规则[8]，将生成的批分配到机器上，即先把批按照加工时间降序排列，然后依照顺序将批安排到首先空闲出来的机器上进行加工。

4.2　算法参数设置

算法参数的设置对 SAGA 的性能有很大的影响。大的种群规模虽然可以得到更好的结果但运行时间过长，而小的种群规模可能不能够充分地覆盖解空间。设置的初始温度 t_0 需要保证算法在迭代过程中能够以合适的概率接收较劣的解，这样就能避免算法陷入局部最优，而合理的降温参数 η 可以控制算法逐步收敛。在使用第一个适应函数迭代 N_1 次后，种群中个体的目标值应该比较接近，然后用第二个适应函数迭代 N_2 次进行加速，算法在迭代 $N(N=N_1+N_2)$ 次后应能收敛到一个近似最优解。合适的 ε 能保证进化过程中最差的个体仍有繁殖的机会，但被选择的概率应小于种群中的其他个体。变异概率 p_m 太大，子代可能会失去从父代继承下来的好特性，p_m 太小就难以得到一些有用的基因序列。实验表明，各参数取如表 2 所示的值时，算法在所有子问题下都能得到较优的解。

表 2　SAGA 的参数设置

M	t_0	η	N	N_1	N_2	ε	p_m
10	100	0.912	60	40	20	0.85	0.01

4.3　实验结果与分析

所有算法 SAGA、GA、BFLPT、FFLPT 以及求解实例下界 LB 的算法均在 VC 6.0 平台下实现。48 类子问题各生成 100 个随机算例。算法 SAGA 与其他 4 个算法测试结果如表 3、4 所示。

表 3　两台机器测试结果

	s_1					s_2					s_3				
	LB	FFLPT	BFLPT	GA	SAGA	LB	FFLPT	BFLPT	GA	SAGA	LB	FFLPT	BFLPT	GA	SAGA
j_1p_1	17.84	18.74	18.66	18.62	18.60	11.27	11.48	11.48	11.48	11.30	22.02	22.39	22.39	22.24	22.16
j_1p_2	34.59	36.47	36.47	36.09	35.93	21.37	21.82	21.82	21.82	21.57	42.11	43.14	43.04	42.93	42.88
j_2p_1	32.02	34.70	34.46	34.31	33.95	19.22	19.67	19.67	19.67	19.47	40.26	42.15	42.00	41.78	41.63
j_2p_2	62.88	68.22	67.78	67.66	67.04	36.58	37.55	37.54	37.54	37.34	78.00	81.30	81.04	80.75	80.42
j_3p_1	78.67	85.37	84.61	84.60	83.98	43.54	44.92	44.82	44.82	44.38	97.19	101.66	101.30	101.30	100.68
j_3p_2	151.04	164.00	162.40	162.40	161.12	84.58	87.18	87.06	87.06	86.31	190.14	198.55	197.98	197.98	197.08
j_4p_1	154.33	164.49	162.99	162.93	161.76	85.42	87.93	87.79	87.79	87.51	194.76	201.64	201.07	201.07	200.03
j_4p_2	296.94	316.44	313.79	313.77	312.03	163.08	168.17	167.98	167.98	166.63	379.06	391.63	390.40	390.40	389.13

表 4　四台机器测试结果

	s_1					s_2					s_3				
	LB	FFLPT	BFLPT	GA	SAGA	LB	FFLPT	BFLPT	GA	SAGA	LB	FFLPT	BFLPT	GA	SAGA
j_1p_1	10.08	10.83	10.80	10.80	10.69	9.03	9.47	9.47	9.47	9.47	12.05	12.23	12.23	12.21	12.18
j_1p_2	20.40	21.21	21.21	21.03	20.99	18.17	18.78	18.78	18.78	18.78	23.06	23.44	23.44	23.38	23.35
j_2p_1	16.66	17.97	17.91	17.84	17.66	10.56	10.68	10.68	10.68	10.62	20.67	21.62	21.55	21.44	21.29
j_2p_2	32.76	35.23	35.05	35.00	34.91	20.47	20.79	20.77	20.77	20.57	40.03	41.72	41.64	41.53	41.48
j_3p_1	39.62	43.02	42.64	42.63	42.34	22.25	22.89	22.84	22.84	22.65	48.84	51.11	50.91	50.91	50.58
j_3p_2	76.07	82.67	81.89	81.89	81.19	43.18	44.50	44.44	44.44	44.10	95.54	99.71	99.47	99.47	99.05
j_4p_1	77.41	82.49	81.74	81.71	81.31	43.06	44.23	44.21	44.21	43.96	97.61	101.06	100.77	100.77	100.20
j_4p_2	148.85	158.56	157.21	157.20	156.61	82.01	84.57	84.47	84.47	83.83	189.85	196.12	195.50	195.50	195.04

在表 3 和 4 中,LB 列表示各类子问题实例下界的平均值,各算法得到的 $C_{\max}$ 的平均值也列在表中,该值越接近下界,表示算法性能越好。从表中可以看出,SAGA 算法得到的结果在 48 类子问题下是最接近下界的。为更清楚地比较算法的性能和变化趋势,选取两台机器下 p_1s_1 类与 p_2s_3 类算例结果绘制算法的性能趋势图分别如图 4 和 5 所示。

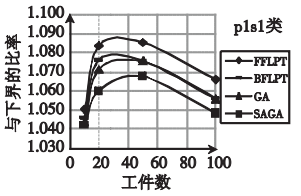

图4　算法的性能及变化趋势(p1s1类)

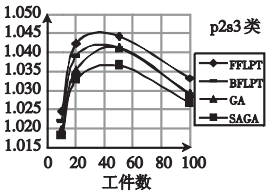

图5　算法的性能及变化趋势(p2s3类)

从图 4 和 5 可知,在工件规模较小的情况下,SAGA 相对于 GA 的优势较弱。这是因为工件规模较小时问题的解空间有限,SAGA 很快收敛,加速适应函数也没有机会发挥其作用,且 GA 在这种情况下也容易找到较优的解。但是随着工件规模的增大,SAGA 的性能优势越来越明显,且逐渐趋于稳定,特别是在工件规模超过 50 之后, GA 与 BFLPT 启发式算法相比几乎没有什么优势,而此时 SAGA 除了改进交叉方式和比 GA 更好的局域搜索能力,同时加速适应函数在进化后期能有效地区别适应度差别较小的个体,因而 SAGA 能够得到更好的结果。

另外,从表 3、4 可以看出,在面对较复杂的问题时,如 $m_2J_4p_2s_2$ 类子问题,这类问题机器多、工件规模大、加工时间较分散且工件尺寸小,该问题的可行解远多于其他问题,而 SAGA 仍然具有明显的优势。说明 SAGA 具有很好的鲁棒性。

5　结束语

本文研究了差异工件平行机批调度问题,以最小化制造时间跨度 $C_{\max}$ 为目标,提出了一种 SAGA。该算法通过将 SA 中的状态转移操作引入基于最优保留的 GA 中,改善了 GA 的局部搜索能力,并能有效避免算法陷入局部最优,同时,在算法迭代后期使用一个加速适应函数进行加速,最后,改进了 GA 的交叉方式。仿真实验结果表明,SAGA 优于传统算法,且具有很好的鲁棒性。批调度问题的研究方向分支很多,进一步的研究可以把 SAGA 推广应用到工件有交货期、工件动态到达或不相容工件族等差异工件平行机批调度问题中,也可以应用到目标为总完工时间 $\sum C_i$ 的问题中。

参考文献:

[1] UZSOY R. Scheduling a single batch processing machine with nonidentical job sizes[J]. International Journal of Production Research, 1994,32(7):1615-1635.

[2] DUPONT L, DHAENENS-FLIPO C. Minimizing the makespan on a batch machine with non-identical job sizes: an exact procedure[J]. Computers & Operations Research, 2002,29(7):807-819.

[3] MELOUK S, DAMODARAN P, CHANG P Y. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing[J]. International Journal of Production Economics, 2004,87(2):141-147.

[4] DAMODARAN P, MANJESHWAR P K, SRIHARI K. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms[J]. International Journal of Production Economics, 2006,103(2): 882-891.

[5] KASHAN A H, KARIMI B, JOLAI F. Minimizing makespan on a single batch processing machine with non-identical job sizes: a hybrid genetic approach[J]. Evolutionary Computation in Combinatorial Optimization, Proceedings, 2006,3906:135-146.

[6] CHANG P Y, DAMODARAN P, MELOUK S. Minimizing makespan on parallel batch processing machines[J]. International Journal of Production Research, 2004,42(19):4211-4220.

[7] KASHAN A H, KARIMI B, JENABI M. A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes[J]. Computers & Operations Research, 2008,35:1084-1098.

[8] DAMODARAN P, CHANG P Y. Heuristics to minimize makespan of parallel batch processing machines[J]. International Journal of Advanced Manufacturing Technology, 2008, 37 (9-10): 1005-1013.

[9] DAMODARAN P, HIRANI N S, VELEZ-GALLEGO M C. Scheduling identical parallel batch processing machines to minimise makespan using genetic algorithms[J]. European Journal of Industrial Engineering, 2009,3(2): 187-206.

[10] MATHIRAJAN M, SIVAKUMAR A I. A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor[J]. International Journal of Advanced Manufacturing Technology, 2006,29(9-10): 990-1001.

[11] POTTS C N, KOVALYOV M Y. Scheduling with batching: a review [J]. European Journal of Operational Research, 2000, 120 (2): 228-249.

[12] RUDOLPH G. Convergence analysis of canonical genetic algorithms [J]. IEEE Trans on Neural Networks, 1994,5(1):96-101.

[13] EIBEN A E, AARTS E H L, VANHEE K M. Global convergence of genetic algorithms: a Markov-chain analysis[J]. Lecture Notes in Computer Science, 1991,496:4-12.

[14] GAAFAR L K, MASOUD S A. Genetic algorithms and simulated annealing for scheduling in agile manufacturing [J]. International Journal of Production Research, 2005,43(14):3069-3085.

[15] YOGESWARAN M, PONNAMBALAM S G, TIWARI M K. An efficient hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS[J]. International Journal of Production Research, 2009,47(19): 5421-5448.

[16] 邢文训, 谢金星. 现代优化计算方法[M].2 版. 北京:清华大学出版社, 2000:141-142.

[17] 王雪梅, 王义和. 模拟退火算法与遗传算法的结合[J]. 计算机学报, 1997,20(4):381-384.