聚类视角下的差异工件平行机批调度问题①

杜 冰, 陈华平, 杨 勃, 李小林
(中国科学技术大学管理学院, 合肥 230026)

摘要: 从聚类角度研究差异工件批调度这一组合优化问题. 论证了差异工件的分批问题实质为一种广义聚类问题, 为求解批调度问题提供了一个全新的途径. 提出了批的空间浪费比的概念, 将最小化批的总加工时间目标变换为最小化批的加权空间浪费比, 从而可以更容易地寻找启发式信息指导分批过程, 两者的等价性也在文中给出了证明. 此外, 以批的空间浪费比为基础, 进一步定义了批间的距离度量, 提出了批的约束凝聚聚类算法 (constrained agglomerative clustering of batches, CACB). 实验结果表明, 与现有的 BFLPT (best-fit longest processing time) 启发式规则和 GA (genetic algorithm) 等算法相比, CACB 在大规模算例的情况下更为有效.

关键词: 调度; 批处理机; 聚类; 组合优化
中图分类号: TP301 文献标识码: A 文章编号: 1007-9807 (2011)12-0027-11

0 引 言

批调度问题是工业工程等领域中新的研究热点. 不同于经典调度问题, 批调度模型中 1 台机器可以同时处理多个工件. 如半导体工业中的芯片预烧工序, 金属加工业的热处理工序, 港口货物装卸等等. 因此, 批调度问题的研究有着极为重要的现实意义.

差异工件批调度问题是批调度中较复杂的一类. 在模型中每个工件具有不同的尺寸, 批中工件的总尺寸不能超过机器的容量限制. Uzsoy[1] 首先提出这个问题, 证明了问题是强 NP 难的, 给出了几个启发式算法, 其中 FFLPT 性能较好. Dupont 等[2] 提出另外两个启发式算法 BFLPT 和 SKP, 及一个分支定界法[3]. 一些学者采用元启发算法求解该问题, 如 Melouk 等[4] 的模拟退火算法, Damodaran[5] 和 Kashan[6] 的遗传算法, 王栓狮等[7] 的蚁群算法, 和程八一等[8] 的粒子群算法.

还有一些研究考虑不同目标函数, 或对模型进行扩展. Sung[9], Ghazvini[10] 和 Chang[11] 研究了目标为总完工时间的批调度问题. Lee 和 Uzsoy[12] 研究了工件动态到达的问题, 但该文中, 工件被假设为单位尺寸以简化模型. Shuguang 等[13] 研究了具有不同到达时间的差异工件批调度问题, 提出近似比为 2 的算法. Chou 等[14] 则采用混合遗传算法求解该问题.

但目前, 对该问题的研究主要集中在单机环境下, 而在实际生产当中, 对批的加工往往是由多台机器[15-17] 同时进行. 因此, 对复杂生产环境下批调度问题的研究显得尤为必要. 本文旨在解决平行机环境下的批调度问题.

1 问题模型

本文研究的问题可具体描述如下.

1) 工件集合 J 中有 n 个待加工的工件, 记为 $J=\{1, 2, \cdots, n\}$. 其中, 工件 j 的加工时间记为 p_j, 尺寸为 s_j.

① 收稿日期: 2009-06-26; 修订日期: 2010-07-19.
基金项目: 创新研究群体科学基金资助项目 (70821001); 博士点基金资助项目 (200803580024).
作者简介: 杜 冰 (1981—), 男, 安徽合肥人, 博士生. Email: toto@mail.ustc.edu.cn

2) 机器集合 M 中有 m 台平行机, 记为 $M=\{1,2,\cdots,m\}$, 其容量均为 C. 设 B 为所有批的集合, B_l ($l\in M$) 为安排在第 l 台机器上加工的批集合. J_b ($b\in B$) 表示批 b 中的工件集合, J_b 不为空集, 且 J_b 中工件尺寸之和不超过 C, 假定没有尺寸大于 C 的工件存在.

3) 批的加工过程不允许中断, 批 b 可以在任意一台机器上加工, 其加工时间 p^b 等于批中工件加工时间的最大值.

4) 第 l 台机器的完工时间记为 T_l ($l=1,2,\cdots,m$), 目标为最小化 $C_{\max}$, 其中 $C_{\max}$ 等于 T_l 的最大值.

根据调度问题的 3 参数表示法, 上述问题可表示为 $Pm\mid batch,\ s_j\leqslant C\mid C_{\max}$, 其数学模型如下

$$\text{Min}\quad C_{\max}=\max_{l\in M}\{T_l\} \tag{1}$$

$$\text{s.t.}\quad \sum_{b\in B}x_{jb}=1,\quad \forall j\in J \tag{2}$$

$$\sum_{l\in M}y_{bl}=1,\quad \forall b\in B \tag{3}$$

$$\sum_{j\in J}s_jx_{jb}\leqslant C,\quad \forall b\in B \tag{4}$$

$$p^b\geqslant p_jx_{jb},\quad \forall j\in J,\ b\in B \tag{5}$$

$$T_l=\sum_{b\in B_l}p^b,\quad \forall l\in M \tag{6}$$

$$x_{jb}\in\{0,1\},\quad \forall j\in J,\ b\in B \tag{7}$$

$$y_{bl}\in\{0,1\},\quad \forall b\in B,\ l\in M \tag{8}$$

$$\left\lceil\sum_{j\in J}\frac{s_j}{C}\right\rceil\leqslant|B|\leqslant n \tag{9}$$

式 (2) 确保每一个工件 j 只能被安排在 1 个批 b 中, 式 (3) 则确保每一批 b 只能被安排在 1 台机器中加工. 式 (4) 保证每个批中的工件尺寸之和不超过机器容量 C. 式 (5) 给出了批加工时间的约束条件. 式 (6) 给出了每台机器的完工时间. 式 (7) 和式 (8) 为决策变量, 若工件 j 被安排在批 b 中, 则 $x_{jb}=1$, 否则 $x_{jb}=0$; 若批 b 被分配到机器 l 上加工, 则 $y_{bl}=1$, 否则 $y_{bl}=0$. $|B|$ 为集合 B 中的元素个数, 即总批数, 范围在 $\left\lceil\sum_{j\in J}\frac{s_j}{C}\right\rceil$ 和 n 之间.

2　批调度问题的聚类算法

2.1　分批问题和聚类问题的异同

批调度问题由两个子问题组成, 一是工件的分批问题; 二是批的加工问题. 分批问题为在满足容量约束的前提下把工件分为多个批次; 批的加工问题是在工件分批后, 如何安排批在机器上的加工序列. 这样, 第 2 个子问题即为经典调度问题 (与经典调度问题不同, 这里批加工时间是不确定的, 由第 1 个子问题给出). 在 $Pm\mid batch,\ s_j\leqslant C\mid C_{\max}$ 模型中, 两个子问题均被证明是 NP 难[1, 18]. 但 Graham[19] 论证了在平行机环境下按照 LPT 规则安排批的加工可以获得很好的性能, 因此 $Pm\mid batch,\ s_j\leqslant C\mid C_{\max}$ 问题的研究重点就在于如何分批.

聚类问题则可表述如下: 给定数据点集合 $X=\{x_1,\cdots,x_N\}$, 其中 $x_i=(x_{i1},x_{i2},\cdots,x_{id})^{\mathrm{T}}\in\Re^d$; 聚类算法将集合中元素划分为 K 个子集 (簇), $\{L_1,\cdots,L_K\}$ ($K\leqslant N$), 且满足

$$L_a\neq\varnothing,\quad a=1,\cdots,K \tag{10}$$

$$\bigcup_{a=1}^{K}L_a=X \tag{11}$$

$$L_a\cap L_b=\varnothing,\quad a,b=1,\cdots,K\ \text{且}\ a\neq b \tag{12}$$

并使某个目标准则最优.

如果将具有二维特征 (加工时间和尺寸) 的工件看做是平面上的数据点, 将工件形成的批看做是数据点的子集 (簇), 则 $Pm\mid batch,\ s_j\leqslant C\mid C_{\max}$ 模型的分批问题实质也是广义聚类问题, 其中式 (10) 是子集非空约束, 在批调度模型中相应的分批也不能为空; 式 (11) 和式 (12) 确保了 K 个子集 $\{L_1,\cdots,L_K\}$ 构成了集合 $\boldsymbol{X}$ 的划分, 这一条件在批调度模型中由式 (2) 和式 (7) 联合保证. 由此可以看出, 上述批调度问题不过是聚类问题的一个特例, 这就使采用聚类方法求解该问题成为一种新的选择.

尽管如此, 通常的聚类算法不能直接用于求解分批问题, 两者还有着明显的差别.

1) 目标函数不同. 传统聚类算法的目标一般是最小化簇内差异, 而最大化簇间差异, 因此, 其目标函数的形式一般为[20]

$$F(X)^{\mathrm{D}}=\min_{P_K\in\pi_K}\|\{D(S_1),\cdots,D(S_K)\}\|_p \tag{13}$$

或

$$F(X)^{\mathrm{R}}=\max_{P_K\in\pi_K}\|\{R(S_a,S_b),\ 1\leqslant a\leqslant b\leqslant K\}\|_p \tag{14}$$

其中, S_a 表示第 a 个簇, $D(S_a)$ 是其直径, $R(S_a, S_b)$ 是簇 a 和簇 b 间的距离或相异度. π_K 表示将对象划分为 K 个子集的所有聚类方案的集合, $P_K = \{S_1, \cdots, S_K\}$ 是聚类的结果, 符号 $\|\cdot\|_p$ 为 l_p 模. 而分批问题的目标是最小化各批总加工时间.

2) 距离或相异度度量不同. 传统聚类问题多用欧氏距离或 Minkowski 距离[21]

$$Distance_{ij} = \left(\sum_{l=1}^{d} |x_{il} - x_{jl}|^n\right)^{1/n} \tag{15}$$

这种距离是静态的, 两个数据点间的距离只跟自身相关. 若两点相同, 则会被划到同一簇中. 而分批问题中工件间的距离不易度量, 两个工件是否被放入同一批, 不仅与两个工件本身相关, 还受到其他工件分批的影响. 即便完全相同的两个工件, 也未必会被放入同一批.

3) 约束条件不同. 传统聚类问题对簇中元素的个数往往没有限制, 而批调度问题由于机器容量约束式 (4) 的存在, 批中工件的个数也受到相应的限制.

因此, 针对分批问题的聚类算法必须为工件找到合适的距离度量, 以适应不同的目标函数. 此外, 在聚类过程中要保证不违反机器容量约束.

2.2　批的空间浪费比的概念

显然, 要使分批方案具有最小的加工时间跨度, 应尽量使每批中的剩余容量较小. 此外, 由于批中工件的加工时间不同, 可能有部分工件已经完成加工但无法交付使用, 该时间被浪费掉. 因此, 使批中冗余时间越小的方案可能越优. 但两个目标之间往往是矛盾的, 一个目标的优化会导致另一个目标变差. 此外, 两者之间没有统一的衡量标准, 难以比较.

为此, 本文提出批的空间浪费比的概念, 并以此为基础, 对聚类算法中的距离进行度量.

定义 1　批 b 中由于工件加工时间差异使机器加工该批时造成的浪费称为 Ⅰ 类浪费空间, 其量纲用工件的加工时间 p 与工件尺寸 s 的乘积来度量. 计算公式为

$$WS_{\mathrm{I}}(b) = \sum_{j\in J_b} (p^b - p_j)s_j \tag{16}$$

定义 2　批 b 中由于工件尺寸之和填不满机器容量 C 使机器加工该批时造成的浪费称为 Ⅱ 类浪费空间, 其量纲也用工件的加工时间 p 与工件尺寸 s 的乘积度量. 计算公式为

$$WS_{\mathrm{II}}(b) = p^b\left(C - \sum_{j\in J_b} s_j\right) \tag{17}$$

定义 3　批 b 中 Ⅰ 类浪费空间与 Ⅱ 类浪费空间之和, 与批中所有工件加工时间和尺寸乘积之和的比率, 称为批 b 的空间浪费比. 计算公式为

$$\begin{aligned} WR(b) &= \frac{WS_{\mathrm{I}}(b) + WS_{\mathrm{II}}(b)}{\sum_{j\in J_b}(p_j s_j)} \\ &= \frac{\sum_{j\in J_b}(p^b - p_j)s_j + p^b\left(C - \sum_{j\in J_b} s_j\right)}{\sum_{j\in J_b}(p_j s_j)} \end{aligned} \tag{18}$$

批的空间浪费比的物理意义是, 在单位空间 (用加工时间 × 尺寸来度量) 的工件被机器加工时, 有多少空间被同时浪费. 由于同时考虑了工件的加工时间和尺寸两个维度的特点, 因此可以更好地作为算法的启发式信息或距离度量. 实际上, 可以得到如下定理.

定理 1　如果一个分批方案将工件集合 J 分为 k 个批, 使得所有批中 Ⅰ 类浪费空间和 Ⅱ 类浪费空间之和最小, 则该方案中批的总加工时间也一定最小.

证明　设 $\Omega = \{P_1, \cdots, P_N\}$ 表示所有可能分批方案的集合. $P_m \in \Omega$ 且 $P_m = \{J_1, \cdots, J_k\}$, 其中 J_b 表示批 b 中的工件集合. 函数 $G(P_m)$ 表示方案 P_m 中所有批的 Ⅰ 类浪费空间和 Ⅱ 类浪费空间之和, 则

$$\begin{aligned} G(P_m) &= \sum_{b=1}^{k}\left(WS_{\mathrm{I}}(b) + WS_{\mathrm{II}}(b)\right) \\ &= \sum_{b=1}^{k}\left(\sum_{j\in J_b}(p^b - p_j)s_j + p^b\left(C - \sum_{j\in J_b} s_j\right)\right) \\ &= \sum_{b=1}^{k}\left(\sum_{j\in J_b}(p^b s_j) - \sum_{j\in J_b}(p_j s_j) + p^b C - p^b\sum_{j\in J_b} s_j\right) = \sum_{b=1}^{k}\left(p^b C - \sum_{j\in J_b}(p_j s_j)\right) \\ &= C\sum_{b=1}^{k} p^b - \sum_{b=1}^{k}\sum_{j\in J_b}(p_j s_j) \end{aligned}$$

于是成立

$$\sum_{b=1}^{k} p^b = \frac{G(P_m) + \sum_{b=1}^{k}\sum_{j\in J_b}(p_j s_j)}{C} \tag{19}$$

式中, C 为给定的机器容量; $\sum_{b=1}^{k}\sum_{j\in J_b}(p_j s_j)$ 为工件集合 J 中所有工件加工时间和尺寸乘积之和. 当工件集合 J 给定, 其值也相应确定. 因此 $\sum_{b=1}^{k}p^b$ 唯一由函数 $G(P_m)$ 决定, 若方案 P_m 使所有批中 Ⅰ 类浪费空间和 Ⅱ 类浪费空间之和最小, 则批的总加工时间也最小. 证毕.

由定理 1, 可以得到如下推论.

推论 **1** 最小化批的总加工时间与最小化批的加权空间浪费比之和也是等价的. 其中, 批 b 空间浪费比的权重为

$$w_b=\frac{\sum_{j\in J_b}(p_j s_j)}{\sum_{b=1}^{k}\sum_{j\in J_b}(p_j s_j)}$$

证明

$$\sum_{b=1}^{k}w_b WR(b)=\sum_{b=1}^{k}\left(\frac{\sum_{j\in J_b}(p_j s_j)}{\sum_{b=1}^{k}\sum_{j\in J_b}(p_j s_j)}\cdot\frac{WS_{\mathrm{I}}(b)+WS_{\mathrm{II}}(b)}{\sum_{j\in J_b}(p_j s_j)}\right)=\sum_{b=1}^{k}\frac{(WS_{\mathrm{I}}(b)+WS_{\mathrm{II}}(b))}{\sum_{b=1}^{k}\sum_{j\in J_b}(p_j s_j)}\tag{20}$$

式中分母 $\sum_{b=1}^{k}\sum_{j\in J_b}(p_j s_j)$ 在 J 给定后为常数, 由定理 1, 最小化 $\sum_{b=1}^{k}(WS_{\mathrm{I}}(b)+WS_{\mathrm{II}}(b))$ 与最小化 $\sum_{b=1}^{k}p^b$ 等价, 故最小化 $\sum_{b=1}^{k}w_b WR(b)$ 也与最小化 $\sum_{b=1}^{k}p^b$ 等价. 证毕.

虽然很难找一个多项式算法使得 $\sum_{b=1}^{k}w_b WR(b)$ 最小, 但若在分批过程中保持较低的空间浪费比, 则方案可能较优. 因为平行机环境下的 $C_{\max}$ 由两个因素决定: 一是分批方案的好坏, 这可以用各批的总加工时间度量; 二是如何安排批在机器上的加工. Graham[19] 论证了按照 LPT 规则安排批在机器上的加工会获得很好的性能, 这是由于 LPT 规则会趋向于平均分配各机器的负载, 尽量减少因机器负载不均衡而造成 $C_{\max}$ 值过大. 因此, 最小化 $C_{\max}$ 主要在于如何分批, 使各批的总加工时间最小. 批的空间浪费比给出了很有用的启发式信息来指导分批过程或是在聚类过程中度量工件间的相异度.

2.3 批的约束凝聚聚类算法 (CACB)

CACB 算法 (constrained agglomerative clustering of batches) 是凝聚式的层次聚类算法, 基本思想是开始时每个工件单独成批, 根据批间的距离度量, 不断地将批合并, 直到满足某个条件时停止. 该算法的关键问题是如何确定批间的距离度量. 由上节可知, 批的空间浪费比与最小化 $\sum_{b=1}^{k}p^b$ 具有强关联性, 因此, 有可能作为批间的距离度量. 假设有两个批 x 和 y, 加工时间 $p^x\geqslant p^y$. 不失一般性, 两批合并时将加工时间少的批 y 加入批 x, 形成一个新批 x'. 但是, 新批 x' 的空间浪费比并不容易计算. 有两种情况.

1) 新批 x' 中不会再次加入其它批, 即批 x' 形成最终分批或此后被加入其它批而消失. 此时, 新批 x' 的 Ⅰ 类浪费空间为

$$WS_{\mathrm{I}}(x')=WS_{\mathrm{I}}(x)+WS_{\mathrm{I}}(y)+|p^x-p^y|s^y\tag{21}$$

式中 $s^y=\sum_{j\in J_y}s_j$ 表示批 y 中工件的尺寸之和.

新批 x' 的 Ⅱ 类浪费空间为

$$WS_{\mathrm{II}}(x')=(C-s^x-s^y)p^x\tag{22}$$

新批 x' 的空间浪费比

$$WR(x')=\frac{WS_{\mathrm{I}}(x')+WS_{\mathrm{II}}(x')}{\sum_{j\in J_x}(p_j s_j)+\sum_{j\in J_y}(p_j s_j)}=\frac{WS_{\mathrm{I}}(x)+WS_{\mathrm{I}}(y)}{\sum_{j\in J_x}(p_j s_j)+\sum_{j\in J_y}(p_j s_j)}+\frac{|p^x-p^y|s^y+(C-s^x-s^y)p^x}{\sum_{j\in J_x}(p_j s_j)+\sum_{j\in J_y}(p_j s_j)}\tag{23}$$

注意到上式第 1 项 $\frac{WS_{\mathrm{I}}(x)+WS_{\mathrm{I}}(y)}{\sum_{j\in J_x}(p_j s_j)+\sum_{j\in J_y}(p_j s_j)}$ 仅仅由批 x 和批 y 本身各自独立决定. 即无论批 x 和批 y 是否合并, $WS_{\mathrm{I}}(x)+WS_{\mathrm{I}}(y)$ 已经存在, 因此应将第 2 项的值作为批间合并的依据.

2) 新批 x' 中会再次加入其它批. 此时情况更

为复杂, 由于批 x' 可能会合并多次, 实际上根本无法精确计算出批 x' 的空间浪费比. 但是, 若有新批加入 x', 则与情况 1) 相比, 批 x' 的 I 类浪费空间将增加, 而 II 类浪费空间将减少, 总的浪费空间也将减少, 并且增加和减少的空间都来自 $(C-s^x-s^y)p^x$ 部分. 这是因为, 该部分代表的是由工件尺寸之和填不满机器容量 C 造成的 II 类浪费空间. 当有新批加入时, 该部分空间会被分割成 3 个部分: 第 1 部分是因为新批的加入而被重新利用的部分; 第 2 部分是因为新批的加入, 由 II 类浪费空间转化为 I 类浪费空间的部分 (由新批中工件加工时间差异造成的); 第 3 部分是由于新批的加入仍然不能填满机器容量, 而残留的 II 类浪费空间. 于是, 引入函数 $E(U,x,y)$, 其中 U 表示当前所有批的集合. $E(U,x,y)$ 根据当前各批的情况, 估算出总浪费空间的减少额所占 $(C-s^x-s^y)p^x$ 的比例, 其值在 [0, 1] 之间.

将情况 1) 和 2) 综合, 可以给出批 x 和批 y 间的距离度量式 (设 $p^x \geqslant p^y$) 为

$$Dist(x,y)=\begin{cases}\dfrac{|p^x-p^y|s^y+(1-E(U,x,y))(C-s^x-s^y)p^x}{\sum\limits_{j\in J_x}(p_j s_j)+\sum\limits_{j\in J_y}(p_j s_j)}, & s^x+s^y\leqslant C\\ +\infty, & s^x+s^y>C\end{cases} \tag{24}$$

如何设计出合适的 $E(U,x,y)$ 是一个有待深入研究的问题, 本文中令

$$E(U,x,y)=\frac{|\{j\ |j\in U-\{x,y\}, s_j\leqslant C-s^x-s^y\}|}{|\{j\ |j\in U-\{x,y\}\}|} \tag{25}$$

$|A|$ 表示集合 A 中元素的个数. 若没有任何批的尺寸小于等于批 x 和 y 合并后的剩余空间, 则 $E(U,x,y)=0$, 即空间 $(C-x^x-s^y)p^x$ 被全部浪费. 若所有批的尺寸都小于批 x 和批 y 合并后的剩余空间, 则 $(C-s^x-s^y)p^x$ 极可能并入新批, 不会被浪费掉, 此时 $E(U,x,y)=1$.

有了距离度量, 可以计算批间距离 $Dist(x,y)$, 并存储在一个二维差别矩阵 DM 中

$$DM=\begin{bmatrix}0 & & & & \\ Dist(2,1) & 0 & & & \\ Dist(3,1) & Dist(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ Dist(k,1) & Dist(k,2) & \cdots & \cdots & 0\end{bmatrix} \tag{26}$$

下一步可找出 DM 中最小的 $Dist(x',y')$, 将 x' 和 y' 合并. 重复该过程, 直到所有批间距离都为 $+\infty$ 停止, 便可得分批方案. 但若工件规模很大, 构建 DM 矩阵和在其中进行查找都会带来相当大的开销. 而每合并 1 次, 批间的动态距离就会根据批的情况发生变化, 需要重新构建 DM 矩阵, 因此其时间复杂度将难以令人接受.

为使算法具有更好的可伸缩性, CACB 算法采取了分而治之的策略. 注意在聚类过程中, 若批 x 和 y 加工时间相差较多, 即 $|p^x-p^y|$ 较大, 则其距离 $Dist(x,y)$ 也较大. 因此, 在聚类的初始阶段, 两批不会合并. 故可以考虑在聚类的第 1 阶段, 将工件集合 J 根据其加工时间划分为若干子集, 对各子集分别聚类, 当矩阵中的最小距离大于某个阈值时停止; 在第 2 阶段, 对第 1 阶段各子集得到的结果进行全局聚类, 得到最终方案. 算法 CACB 的伪码如下.

算法　CACB

输入　工件集合 J; 机器容量 C; 分割参数 ρ; 阈值参数 θ

输出　批集合 U 及批的总加工时间 $\sum_{b\in U}p^b$.

步骤**1**　按照加工时间, 将工件集合 J 均匀地划分为 ρ 个子集.

步骤**2**　将每个工件单独放入一批, 得到 ρ 个批集合: $U_1, U_2, \cdots, U_\rho$.

步骤 **3**　for $i=1$ *to* ρ

步骤 **4**　对集合 U_i 中的批, 根据式 (24) 计算距离 $Dist(x,y)$, 并存储在差别矩阵 DM_i 中.

步骤 **5**　找出 DM_i 中最小距离 $Dist(x',y')$, 若 $Dist(x',y')\leqslant\theta$ 则合并批 x' 与批 y'; 否则转步骤 7.

步骤 **6**　更新集合 U_i 中的批, 转步骤 4.

步骤 **7**　end for

步骤 **8**　令集合 $U=U_1\cup U_2\cdots\cup U_\rho$

步骤 **9**　对集合 U 中的批, 根据式 (24) 计算

距离 $Dist(x, y)$, 并存储在差别矩阵 DM 中.

步骤 **10** 找出 DM 中最小距离 $Dist(x', y')$, 若不为 $+\infty$, 则合并批 x' 与批 y'; 否则转步骤 12.

步骤 **11** 更新集合 U 中的批, 转步骤 9.

步骤 **12** 计算 $\sum_{b\in U} p^b$, 输出集合 U.

2.4 基于 LPT 规则安排批的加工

在获得分批后, 接下来是如何安排批在机器上的加工. Pinedo[18] 证明, 当机器数量 $m=2$ 时, 安排批在机器上的加工问题等价于二分问题. 由于二分问题是NP难的, 故而安排批在平行机上的加工也是 NP 难的. 但是, Graham[19] 论证了在平行机环境下将批按照LPT规则排序安排加工可以获得非常好的性能, 步骤如下: 将所有批按照加工时间降序排列. 选取队列中第 1 个批, 当有机器空闲时, 立刻将批送入该机器加工, 直到所有批都分配完毕.

3 仿真实验

3.1 实验简述

本文采用 Melouk 等[4] 提出的方法生成测试算例. 考虑了问题在机器台数, 工件规模, 工件加工时间及尺寸 4 个维度的变化. 各维度等级划分及取值范围见表 1. 为更好地检验各算法的可伸缩性, 在通常 100 个工件为上限的基础上, 增加测试了 200 和 300 工件的情形. 每类问题可用 $M_aJ_bp_cs_d$ ($a=1,2$; $b=1,2,\cdots,6$; $c=1,2$; $d=1,2,3$) 表示. 例如, 2 台机器下, 工件规模为20, 工件加工时间服从 [1, 10] 离散均匀分布, 工件尺寸服从 [4, 8] 离散均匀分布的实例表示为 $M2J2p1s3$. 实验中总计测试了 72 类不同子问题, 所有实例中假设机器容量 C 都为 10.

实验中测试的算法包括 CACB, Damodaran 等[5] 提出的 GA 以及目前已知最好的启发式算法 BFLPT. 此外, 按如下方法[17] 计算出问题下界: 令 $Q=\{j \mid C-s_j<\min\{s_1,\cdots,s_n\}, j\in J\}$, 则集合 Q 中的工件只能单独成批, 先计算出 Q 中工件加工时间之和. 然后, 对集合 $J-Q$ 中的工件进行松弛, 即加工时间 p_j, 尺寸为 s_j 的工件被松弛为 s_j 个加工时间为 p_j 的单位尺寸工件. 对松弛后的工件按加工时间降序排列, 然后连续 C 个工件组成一批, 可得下界为

$$C_{\max}^{LB}=\max\left\{\frac{\sum_{i\in Q}p_i+C_{J-Q}^{\text{relax}}}{m}, \max_{i=1,\cdots,n}\{p_i\}\right\} \quad (27)$$

式中 C_{J-Q}^{relax} 为松弛问题中所有批的加工时间之和.

表 1 问题因素及等级划分

Table 1 Factors and levels

维度	等级划分
机器台数: $M1-M2$	2, 4
工件规模: $J1-J6$	10, 20, 50, 100, 200, 300
工件加工时间: $p1-p2$	离散均匀分布 [1, 10], 离散均匀分布 [1, 20]
工件尺寸: $s1-s3$	离散均匀分布 [1, 10] (混合工件集), 离散均匀分布 [2, 4] (小工件集), 离散均匀分布 [4, 8] (大工件集)

3.2 算法参数设置

算法 CACB 包含两个参数, 分割参数 ρ 和阈值参数 θ.

ρ 控制第 1 阶段聚类中子集的数量, ρ 越大则子集的工件数越少, 算法效率高, 但同时全局搜索能力也随之降低. ρ 根据工件规模和加工时间而定. 规模越大, 加工时间差异越大, 则 ρ 越高, 反之则越低.

θ 控制批合并时的距离, 其值过小, 则在第 1 阶段聚类时只有形成很小浪费空间的两个批才被合并, 这会造成第 2 阶段聚类的计算负担; 其值太大, 则会在聚类的过程中造成较大的浪费空间. θ 的设置需综合考虑工件规模、尺寸、加工时间以及分割参数 ρ. 第 1 阶段聚类的子集中工件数较多, 工件加工时间和尺寸差异越小, θ 可以设置得越小, 反之则越大.

GA 算法的参数设置依据原文献 [5], 种群规模 $pop=20$, 变异概率 $pm=0.01$. 由图 1 知, 工件数较少的情况下算法收敛较快, 而当工件规模达到 300 时, 需要超过 100 次迭代算法才会逐渐收敛. 因此, 将算例中 GA 的迭代次数设为 100.

表 2　CACB 的参数设置

Table 2 Parameter settings for CACB

	$p1s1$	$p1s2$	$p1s3$	$p2s1$	$p2s2$	$p2s3$
$J1$	$\rho=1,\ \theta=0$	$\rho=1,\ \theta=0$	$\rho=1,\ \theta=0$	$\rho=1,\ \theta=0$	$\rho=1,\ \theta=0$	$\rho=1,\ \theta=0$
$J2$	$\rho=1,\ \theta=0$	$\rho=1,\ \theta=0$	$\rho=1,\ \theta=0$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.2$
$J3$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.2$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.2$	$\rho=2,\ \theta=0.2$
$J4$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.2$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.1$	$\rho=2,\ \theta=0.3$
$J5$	$\rho=3,\ \theta=0.2$	$\rho=3,\ \theta=0.1$	$\rho=3,\ \theta=0.2$	$\rho=4,\ \theta=0.2$	$\rho=4,\ \theta=0.1$	$\rho=4,\ \theta=0.2$
$J6$	$\rho=5,\ \theta=0.2$	$\rho=5,\ \theta=0.1$	$\rho=5,\ \theta=0.2$	$\rho=6,\ \theta=0.2$	$\rho=6,\ \theta=0.1$	$\rho=6,\ \theta=0.3$

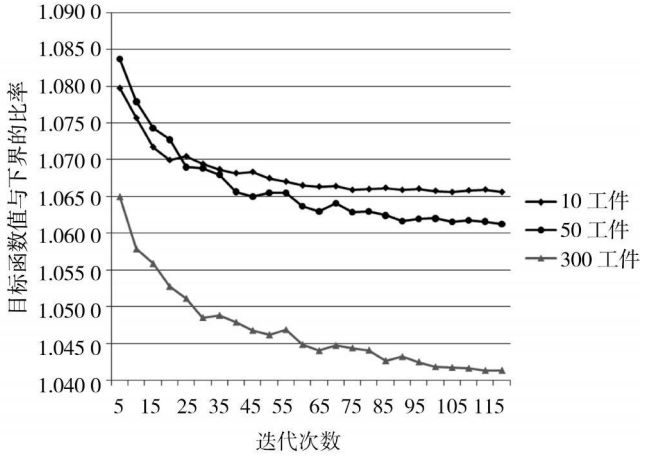

图 1　GA 算法的收敛趋势

Fig. 1 Convergence trend for GA

3.3　实验结果及分析

所有算法 CACB, GA, BFLPT 及求解实例下界 LB 的算法均在 Visual C ++2005 平台上编程实现. 72 类子问题各生成了 100 个随机算例, 测试结果如下.

表 3　2 台机器测试结果

Table 3 Experimental results on 2 machines

	$s1$				$s2$				$s3$			
	LB	BFLPT	GA	CACB	LB	BFLPT	GA	CACB	LB	BFLPT	GA	CACB
$J1p1$	17.46	18.75	18.61	19.10	10.41	11.22	11.12	11.38	20.87	21.34	21.32	21.42
$J1p2$	35.23	37.61	37.51	38.21	20.13	21.96	21.99	23.37	40.94	42.25	42.17	42.43
$J2p1$	32.03	34.47	34.39	34.76	18.94	19.68	19.55	20.34	41.25	42.50	42.34	42.64
$J2p2$	62.94	67.95	67.83	68.91	35.80	37.53	37.50	39.46	78.69	81.48	81.38	81.72
$J3p1$	77.54	82.50	82.31	82.94	44.10	45.32	45.12	46.21	98.68	102.93	102.82	102.65
$J3p2$	146.77	156.49	156.43	157.22	84.49	87.29	87.25	90.73	188.98	196.01	195.96	195.41
$J4p1$	152.94	161.75	161.49	161.56	84.91	87.45	87.11	87.87	192.73	199.28	198.87	198.30
$J4p2$	289.66	307.69	307.62	306.77	161.25	166.34	166.37	169.93	372.10	384.02	384.07	381.93
$J5p1$	309.56	324.20	324.84	322.13	167.26	172.26	172.06	171.36	389.66	399.52	398.80	397.30
$J5p2$	580.60	606.60	607.18	604.09	318.79	328.31	328.36	327.38	739.30	757.42	757.40	753.61
$J6p1$	457.23	475.77	476.39	471.55	249.68	257.32	257.48	255.25	579.01	590.96	590.32	587.34
$J6p2$	867.61	902.65	903.04	898.82	478.26	492.25	492.95	490.94	1 106.48	1 130.37	1 131.26	1 124.71

表4　4 台机器测试结果

Table 4 Experimental results on 4 machines

	*s*1				*s*2				*s*3			
	LB	BFLPT	GA	CACB	LB	BFLPT	GA	CACB	LB	BFLPT	GA	CACB
*J*1*p*1	9.13	10.93	10.87	11.07	5.66	9.57	9.57	9.57	11.10	12.25	12.21	12.32
*J*1*p*2	17.08	20.70	20.64	20.86	10.46	18.53	18.53	18.53	20.87	23.27	23.18	23.43
*J*2*p*1	16.88	18.55	18.51	18.65	9.74	10.72	10.72	11.41	20.63	21.85	21.71	21.72
*J*2*p*2	31.92	35.37	35.25	35.76	18.10	20.78	20.77	21.77	38.64	40.88	40.70	40.96
*J*3*p*1	39.30	42.12	42.13	42.04	22.19	22.95	22.91	23.94	50.15	52.23	52.16	52.00
*J*3*p*2	74.97	80.46	80.38	80.73	41.33	43.27	43.25	45.53	95.23	99.03	98.97	98.89
*J*4*p*1	76.35	80.63	80.62	80.30	42.70	44.06	44.04	44.39	97.43	100.47	100.45	99.91
*J*4*p*2	147.21	155.52	155.54	155.68	81.07	83.49	83.57	85.55	186.27	192.03	192.02	191.24
*J*5*p*1	151.30	157.99	158.04	156.82	84.13	86.76	86.77	86.47	194.03	199.35	199.31	198.31
*J*5*p*2	290.58	303.66	303.70	302.44	158.47	163.05	163.22	163.72	368.92	378.33	378.32	376.73
*J*6*p*1	227.77	236.45	236.52	234.62	125.41	129.31	129.39	128.27	290.69	297.03	297.02	295.46
*J*6*p*2	437.52	454.80	452.52	454.84	239.72	246.84	246.82	245.79	552.34	564.37	564.39	561.46

表中可以看出，随着工件规模的增大，本文提出的 CACB 算法性能优势愈加明显. 为更清楚比较算法的性能及变化趋势，选取 2 台机器下 *p*1 类算例绘制性能趋势图，见图 2.

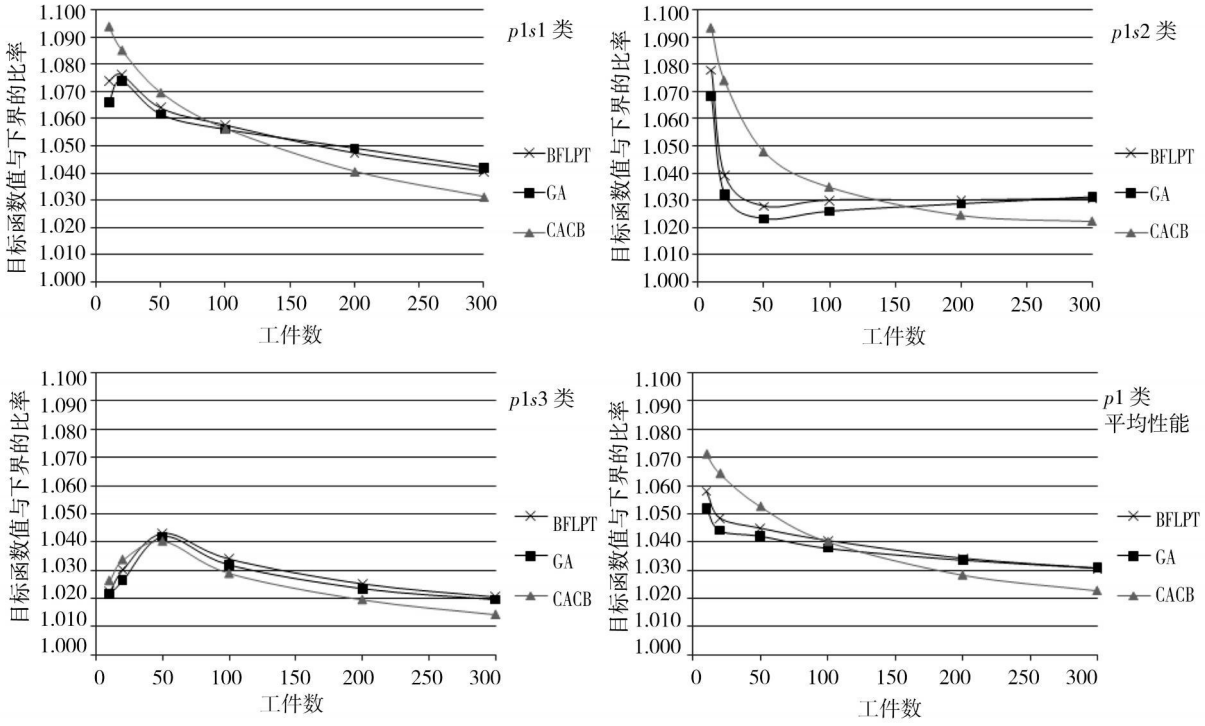

图2　算法性能变化趋势

Fig.2 Performance trends

从上图可知，在工件规模较小的情况下，CACB 相对 GA 并没有什么优势. 这是因为 CACB 是一种非回溯算法，一旦两个批被合并，则将不会拆开，即便该合并决定是错误的，在工件数量少的情况下一个错误的合并对目标函数值影响很大. 而 GA 可以不断地迭代寻优，在工件数为 10 和 20 的小规模算例下有着不错的性能，但随着工件规模的增大，GA 与下界的比例很快升高. 因为文

献[5] 提出的这种 GA 算法是按工件序列的方式进行编码, 然后依 BF 规则来生成批的, 其搜索空间规模为 $n!$. 而在不考虑约束的情况下, 将 n 个工件放入 k 个批的方案总数为[22]

$$P(n,k)=\frac{1}{k!}\sum_{m=1}^{k}(-1)^{k-m}C_k^m m^n \qquad (28)$$

由于批数 k 可能是从 1 到 n, 故总的解空间规模为 $\sum_{k=1}^{n}P(n,k)$, 在 n 很大的情况下, 其值远高于 $n!$, 故该 GA 只能在小的一部分解空间进行搜索. 另外, 算法性能还受分批规则影响.

与启发式算法 BFLPT 相比, CACB 的优势则更为明显. 因为 BFLPT 每一步只考虑工件序列中最前面的工件, 并不考察其后工件的情况. 而 CACB 在做出合并决定之前会检查集合中所有批的情况, 选择可能造成最小浪费的两个批合并.

图中可以看出, 各算法性能在大工件情况, 即 $s3$ 类子问题中较为接近且变化平稳. 这是由于 $s3$ 类问题工件服从 [4, 8] 的离散均匀分布, 由于机器容量为 10, 故每批最多只能包含两个工件, 尺寸为 7, 8 的工件只能单独成批, 只需要为尺寸 4, 5, 6 的工件决定分批, 因此问题的复杂度大大降低了. 各算法与下界的比例均在 1. 01 到 1. 05 之间. 相反, 在小工件 $s2$ 类问题中, 问题更为复杂, 解空间更大, 各算法的性能差异和波动都比较大.

此外, 为了了解各算法在不同类型算例下的时间性能, 表5 列出了 2 台机器下各算法在每个子类问题中运行 100 个算例的总时间.

表5　算法运行时间比较

Table 5 Runtime comparison of algorithms

	$s1$			$s2$			$s3$		
	BFLPT /s	GA /s	CACB /s	BFLPT /s	GA /s	CACB /s	BFLPT /s	GA /s	CACB /s
$J1p1$	0.078 1	6.859 4	0.062 5	0.031 3	6.140 6	0.046 9	0.060 5	7.062 5	0.078 1
$J1p2$	0.109 4	7.012 5	0.031 3	0.015 6	6.109 4	0.031 3	0.031 3	7.453 1	0.062 5
$J2p1$	0.078 1	11.875 0	0.125 0	0.046 9	9.812 5	0.140 6	0.203 1	11.781 3	0.067 5
$J2p2$	0.098 1	11.500 0	0.156 3	0.031 3	10.234 4	0.265 6	0.062 5	12.187 5	0.187 5
$J3p1$	0.359 4	31.562 5	0.718 8	0.140 6	25.234 4	1.156 3	0.656 3	33.437 5	0.609 4
$J3p2$	0.218 8	32.250 0	0.546 9	0.093 8	24.937 5	0.968 8	0.546 9	34.781 3	0.671 9
$J4p1$	1.281 3	82.625 0	2.921 9	0.500 0	61.375 0	7.609 4	1.796 9	93.046 9	2.328 1
$J4p2$	1.078 1	82.687 5	3.562 5	0.625 0	61.937 5	7.265 6	1.796 9	91.687 5	2.265 6
$J5p1$	4.250 0	250.734 4	18.531 3	1.515 6	178.578 1	104.718 8	6.875 0	285.921 9	9.812 5
$J5p2$	4.109 4	241.609 4	21.156 3	1.265 6	170.453 1	101.796 9	6.296 9	267.984 4	9.953 1
$J6p1$	8.953 1	475.515 6	66.328 1	3.125 0	334.093 8	362.343 8	17.656 3	676.906 3	31.250 0
$J6p2$	10.859 4	544.546 9	78.546 9	3.281 3	331.218 8	349.859 4	12.984 4	512.828 1	23.250 0

由表中可以看出, 算法 BFLPT 的运行时间最短. 因为该算法是一种简单的启发式规则, 本质是贪心算法, 每一步仅考虑当前最优, 故而速度极快. 但这可能会造成在一些特殊情况下, 算法的性能会非常差, 已有一些文献[2-3] 对此进行了研究. 而对比 GA, CACB 在绝大部分算例上运行时间都要少得多, 尤其在大工件情况, 因为此时每个批能容纳的工件较少, 批数则较多, CACB 对批的合并次数也随之减少.

总体而言, 无论是大工件、小工件还是混合工件的算例, 本文提出的 CACB 算法随着工件规模的增加而表现出了良好的性能, 适于应用到生产实践中.

4　结束语

本文从聚类角度研究了差异工件批调度这类组合优化问题, 为求解该问题提供了全新的思路. 提出了批的空间浪费比的概念, 并证明了最小化批的加权空间浪费比之和与最小化批的总加工时间是等价的, 从而使得寻找与目标函数强相关的启发式信息变得更加直观和容易. 在批的空间浪

费比概念的基础上进一步定义了批间的距离，提出了批的约束凝聚聚类算法 CACB. 实验结果表明，该算法在大规模工件的情况下比现有的启发式算法 BFLPT 和 GA 算法更有效.

进一步的研究可以从以下几个方面进行：一是改进 CACB 中的聚类方式，设计出更高效的聚类算法. 二是将批的空间浪费比概念与微粒群算法和蚁群算法等群智能算法相结合，作为这些算法中的启发式信息，提高求解效率. 三是将聚类模型推广到 3 维空间，即应用到工件具有动态到达时间的批调度问题中. 四是从更一般的角度考虑，是否可以从数据挖掘的角度来求解不同类型的组合优化问题，以及如何设计出通用的模型框架.

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, Ghazvini F J. Minimizing makespan on a single batch processing machine with non-identical job sizes[J]. European Journal of Automation(JESA), 1998, 32(4): 431 – 440.

[3] 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.

[4] 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.

[5] 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.

[6] Kashan A H, Karimi B, Jolai F. Minimizing makespan on a single batch processing machine with non-identical job sizes: A hybrid genetic approach[C]//Evolutionary Computation in Combinatorial Optimization, Proceedings, Berlin: Springer-Verlag, 2006.

[7] 王栓狮，陈华平，程八一，等. 一种差异工件单机批调度问题的蚁群优化算法[J]. 管理科学学报，2009，12(6)：72 – 82.
Wang Shuan-shi, Chen Hua-ping, Cheng Ba-yi, et al. Minimizing makespan on a single batch processing machine with non-identical job sizes using ant colony optimzation[J]. Journal of Management Sciences in China, 2009, 12(6): 72 – 82. (in Chinese)

[8] 程八一，陈华平，王栓狮. 模糊制造系统中的不同尺寸工件单机批调度优化[J]. 计算机集成制造系统，2008，14(07)：1322 – 1328.
Cheng Ba-yi, Chen Hua-ping, Wang Shuan-shi. Optimization for scheduling a single batch-processing machine with non-identical job sizes in fuzzy manufacturing system[J]. Computer Integrated Manufacturing Systems, 2008, 14(07): 1322 – 1328. (in Chinese)

[9] Sung C S, Joo U G. Batching to minimize weighted mean flow time on a single machine with batch size restrictions[J]. Computers & Industrial Engineering, 1997, 32(2): 333 – 340.

[10] Ghazvini F J, Dupont L. Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes[J]. International Journal of Production Economics, 1998, 55(3): 273 – 280.

[11] Chang P C, Wang H M. A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes[J]. International Journal of Advanced Manufacturing Technology, 2004, 24(7/8): 615 – 620.

[12] Lee C Y, Uzsoy R. Minimizing makespan on a single batch processing machine with dynamic job arrivals[J]. International Journal of Production Research, 1999, 37(1): 219 – 236.

[13] Shuguang L, Guojun L, Xiaoli W, et al. Minimizing makespan on a single batching machine with release times and non-identical job sizes[J]. Operations Research Letters, 2005, 33(2): 157 – 164.

[14] Chou F D, Chang P C, Wang H M. A hybrid genetic algorithm to minimize makespan for the single batch machine dynamic scheduling problem[J]. International Journal of Advanced Manufacturing Technology, 2006, 31(3/4): 350 – 359.

[15] 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.

[16] 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.

[17] 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(4): 1084 – 1098.

[18] Pinedo M. Scheduling: Theory, Algorithms and Systems[M]. New York: Springer, 2008.

[19] Graham R. Bounds on multiprocessing timing anomalies[J]. SIAM Journal on Applied Mathematics, 1969, 17(2): 416 – 429.

[20] Pinter J, Pesti G. Set partition by globally optimized cluster seed points[J]. European Journal of Operational Research, 1991, 51(1): 127 – 135.

[21] Rui X, Wunsch D II. Survey of clustering algorithms[J]. IEEE Transactions on Neural Networks, 2005, 16(3): 645 – 678.

[22] Liu C. Introduction to Combinatorial Mathematics[M]. New York: McGraw-Hill, 1968.

Scheduling parallel batching machines with non-identical job sizes from a clustering perspective

DU Bing, CHEN Hua-ping, YANG Bo, LI Xiao-lin

School of Management, University of Science and Technology of China, Hefei 230026, China

Abstract: The problem of scheduling parallel batch processing machines is considered from a clustering perspective in this paper. We first demonstrate that the batching problem with non-identical job sizes can be regarded as a generalized clustering problem, providing a novel insight into scheduling with batching. The concept of WR (waste ratio of batch space) is then presented and the objective function of minimizing makespan is transformed into minimizing weighted WR so as to define the distance measure between batches in a more understandable way. The equivalence of the two objective functions is also proved. In addition, a clustering algorithm CACB (constrained agglomerative clustering of batches) is proposed based on the definition of WR to generate batches. The experimental results show that CACB outperforms the existing approaches BFLPT (best-Fit longest processing time) and GA (genetic algorithm) in large-scale problems.

Key words: scheduling; batch processing machine; clustering; combinatorial optimization