计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(1):162−167 [doi: 10.15888/j.cnki.csa.006136]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

蚁群-鱼群混合算法在差异工件批调度中的应用①

吕顺风, 马　科

(中国科学技术大学 管理学院, 合肥 230026)

摘　要: 调度问题是组合优化领域中一类重要的问题, 批调度问题更是考虑了工件的尺寸和机器的容量, 增加了调度的难度. 本文针对差异工件批调度问题, 把蚁群算法和鱼群算法相结合, 提出了一种混合算法: 引入鱼群算法中拥挤度的概念, 并且与蚁群算法相结合, 这不仅能避免算法早熟现象的发生, 也加快了算法后期的收敛速度. 通过负载率与利用率的比较, 混合算法相对于单一的算法, 有着更高的效率和更好的效果, 能够使寻优个体更快的寻找到满意解.

关键词: 蚁群算法; 拥挤度; 批调度

引用格式: 吕顺风,马科.蚁群-鱼群混合算法在差异工件批调度中的应用.计算机系统应用,2018,27(1):162–167. http://www.c-s-a.org.cn/1003-3254/6136.html

Application of Ant Colony Hybrid Algorithm in Batch Scheduling of Differentiated Jobs

LV Shun-Feng, MA Ke

(The School of Management, University of Science and Technology of China, Hefei 230026, China)

Abstract: Scheduling is an important issue in the field of portfolio optimization, and batch scheduling is more complicated since it takes into consideration of workpieces sizes and machine capacity. In this study, to solve the batch scheduling problems with non-identical sizes, we propose a hybrid algorithm based on ant colony algorithm and fish swarm algorithm. By introducing the fish algorithm's swarm degree into the ant colony algorithm, the hybrid algorithm does not only avoid the premature, but also accelerates the convergence speed of the algorithm. In respect of load rate and utilization, the optimization algorithm has higher efficiency and achieves better results, because it can reduce searching time in finding optimal solution.

Key words: ant colony algorithm; swarm degree; batch scheduling

1　引言

调度问题是组合优化领域中一类重要的问题, 在生产管理、通信等方面有着重要作用. 批调度问题是其中的一个重要的分支, 而且与经典调度问题不同的是: 一台机器可以同时处理多个工件; 批的加工时间等于批中工件的最大完成时间; 批的完成时间等于批的开始时间加上批的最大加工时间; 批的完成时间等于批中工件的最大完成时间. 批调度问题考虑了工件的尺寸和机器的容量, 这在实际的活动中经常需要考虑到, 例如半导体芯片的预烧、电路测试、港口货物装卸、金属加工等等. 这些问题中, 工件的尺寸是有差异的, 及其加工需要分批进行, 批的尺寸不能超过机器的容量. 这类问题是经典调度问题在空间上的拓展, 也是生产调度领域中一个新的研究方向.

1.1　问题的描述

针对此类问题, 首先作如下假设[1]:

(1) 有 n 个待加工的工件{1, 2,…, n}, 工件的加工时间为 p_j, 工件的尺寸为 s_j;

① 基金项目: 国家自然科学基金 (71671168)
收稿时间: 2017-04-01; 修改时间: 2017-04-20; 采用时间: 2017-04-26; csa 在线出版时间: 2017-12-22

(2) 机器的容量为 B, 批中工件的总尺寸都小于 B, 假设没有工件尺寸大于 B 的工件;

(3) 假设批一旦开始加工, 在批加工完成之前, 不可以被打断或者添加新的工件, 批 k 的加工时间为 T_k, 等于批中最大加工时间的工件;

(4) 目标是最大加工时间 $C_{\max}$ 最小, 其中最大加工时间等于批中最后一个离开机器的工件完成时间, NSBM 问题的制造跨度为所有批的加工时间总和.

该问题采用三参数表示法可以描述为$1\left|B,s_j\right|C_{\max}$, 数学模型如下:

$$Min\ C_{\max}=\sum_{i=1}^{k}T_k \tag{1}$$

$$\text{s.t.}\ \sum_{i=1}^{k}x_{ij}=1\quad i=1,2,...,n \tag{2}$$

$$\sum_{i=1}^{n}s_ix_{ij}\le B\quad j=1,2,...,k \tag{3}$$

$$p_ix_{ij}\le T_j\quad i=1,2...n,\ \ j=1,2,...,k \tag{4}$$

$$0\le T_j \tag{5}$$

$$x_{ij}\in\{0,1\}\quad i=1,2,...,n,\ \ j=1,2,...,k \tag{6}$$

$$\sum_{i=1}^{n}\frac{s_i}{B}\le k\le n\quad k\in Z^* \tag{7}$$

其中, k 是分批的批数, x_{ij} 是 0–1 变量, 式 (1) 表示优化的目标是制造跨度; 式 (2) 限定每个工件只能分到一个批次中; 式 (3) 表示机器容量约束; 式 (4) 表示批加工时间; 式 (5)、式 (6) 和式 (7) 是基本约束.

Uzsoy 首先提出了该类问题的单机器模型, 即工件尺寸不同的单机批调度问题 (single batch-processing machine with non-identical job sizes, NSBM), 证明了当所有工件加工时间相同时, 问题$1\left|B,s_j\right|C_{\max}$等价于容量为 B 物品尺寸为 s_i 的装箱问题. 由于装箱问题是强 NP 难的, 所以$1\left|B,s_j\right|C_{\max}$问题也是强 NP 难的[2].

针对问题$1\left|B,s_j\right|C_{\max}$, Uzsoy 根据装箱问题的 first-fit 的启发式规则提出了几个启发式算法, 其中 FFLPT 性能最好; Dupont 给出了另外两种启发式算法 BFLPT 和 SKP(successive knapsack problem); Dupont 和 Flipo 提出了问题的分支定界法; Melouk 和 Damodaran 分别利用模拟退火算法 (SA)[3]和遗传算法 (GA)[4]求解了此问题, 但是并没有和性能较好的启发式算法相比较.

2 蚁群算法和鱼群算法

由于在构建差异工件 (即工件有尺寸差异) 单机批调度问题的解时, 批的加工时间是不确定的, 从而不能类似于经典调度问题的蚁群算法, 把批加工时间的倒数作为蚁群算法中的启发式信息. 针对这一问题, 我们引入批的利用率和批的负载均衡率作为蚁群算法中的启发式信息, 提出了 JACO (ant colony optimization based a job sequence) 和 BACO (ant colony optimization based a batch sequence) 两种蚁群优化算法[5].

2.1 启发式信息

在一组给定的批次下, 当批的总剩余空间越小, 方案越佳. 此外, 批的加工时间等于批中加工时间的最大值, 如果在加工完之前, 有的工件已经加工完毕却没有办法交付使用, 那么这段浪费掉的时间我们称之为冗余时间. 很显然, 对于白白浪费掉的时间, 冗余时间越小的方案往往是最优的方案.

批的利用率为批的加工尺寸所占机器容量的比例, 公式为:

$$\mathrm{UR}_k=\sum_{i\in B_k}s_i$$

批的负载率为批中加工时间的变异系数与 1 的差值, 公式为:

$$BR_k=1-\frac{\sigma_k}{\omega_k}=1-\frac{\sqrt{\frac{1}{|B_k|}\sum_{i\in B_k}(p_i-\omega_k)^2}}{\omega_k}$$

W_k 为加工时间的平均值, σ_k 表示批加工时间的方差, 批的负载率越大, 表示批中加工时间分布越均匀, 加工方案越好[6].

2.2 算法的描述

2.2.1 JACO 蚁群算法

在本文中, 我们采用 JACO 蚁群算法算法, 直接考虑工件序列, 以方便算法的比较. 具体描述如下[7]:

(1) 信息素的定义: τ_{ij} 表示工件 j 排列在工件 i 之后的信息素浓度, 分批的规则采用 BF (Best Fit) 规则.

(2) 启发式信息: 利用 BF 规则进行分批, 实际上就是利用批利用率最大这个启发式信息, 在生成工件序列后, 利用 BF 规则进行分批, 工件中间隔较小的工件被分在同一批中概率较大. 为方便计算, 定义工件 j 排列在工件 i 之后的启发式信息为:

$$\eta_{ij}=\frac{1}{1+|p_i-p_j|}$$

(3) 状态转移概率公式记为 $allowed_a$={j|工件 j 未被 a 访问到}, 蚂蚁 a 当前位置工件 i, 那么它的状态转

移选择下一个工件概率为[8]:

$$p_{i,j}^{a}=\begin{cases}\dfrac{\tau_{i,j}^{a}\eta_{i,j}^{\beta}}{\sum_{\notin allowed_a}\tau_{i,j}^{a}\eta_{i,j}^{\beta}} & j\in allwoed_a\\ 0 & j\notin allwoed_a\end{cases} \tag{8}$$

(4) 信息素的更新:

$$\begin{aligned}&\tau_{i,j}(t+1)=(1-\rho)\tau_{i,j}(t)+\Delta\tau_{i,j},\\&\Delta\tau_{i,j}=\sum_{1}^{m}\Delta\tau_{i,j}^{a}\end{aligned} \tag{9}$$

2.2.2 鱼群算法

人工鱼个体的状态可定义为向量$X=(x_1,x_2,\ldots,x_n)$, 其中$x_i(i=1,\ldots,n)$为寻优的变量;人工鱼当前所在的位置的食物浓度定义为 $Y=f(x)$, 其中 Y 是目标函数值; 人工鱼的个体之间的距离定义为$d_{i,j}=\left\|X_i-X_j\right\|$; 人工鱼的视野定义为 Viusal, 表示人工鱼可以感知的范围;人工鱼的移动步长定义为 Step, 表示人工鱼在视野范围内, 找到一个更优点, 每向该方向移动一次单位距离, Step 增加 1; 拥挤度因子定义为 δ, 表示在一定范围内的能够容纳人工鱼数量的限制, 防止食物充足的地方聚集过多的人工鱼[9,10].

(1) 觅食行为 (prey)

觅食行为的行为选择为:

$$\begin{cases}X_i\rightarrow X_j & Y_i<Y_j, d_{i,j}<visual\\ X_i\rightarrow random & \text{else}\end{cases}$$

(2) 聚群行为 (swarm)

聚群行为的行为选择为:

$$\begin{cases}X_i\rightarrow X_c & Y_i<Y_c, d_{i,c}<visual, \dfrac{n_f}{N}<\sigma\\ X_i\rightarrow prey & \text{else}\end{cases}$$

X_c 表示领域内伙伴的中心位置.

(3) 追尾行为 (follow)

追尾行为的行为选择:

$$\begin{cases}X_i\rightarrow X_j & Y_j=\max\{Y_n\}, d_{i,j}<visual, \dfrac{n_f}{N}<\sigma\\ X_i\rightarrow prey & \text{else}\end{cases}$$

3 混合算法

3.1 蚁群算法和鱼群算法的优化机理

蚁群算法和鱼群算法虽然都属于群体优化算法, 但是算法中的个体并没有表现出智能性, 个体的行为简单, 只具有基础的判断能力, 无法智能的遵循相应行为规则, 但是当它们形成一个群体的时候, 整个群体会表现出一定的智能性. 这两种算法的个体规律有一定的相似性, 人工鱼群算法初期, 人工鱼的觅食行为是随机的, 这和蚁群算法是相似的; 人工鱼群算法后期, 人工鱼的聚集是由伙伴中心的最优状态决定的, 这与蚁群算法中信息素浓度和启发式信息大小有着相同的作用. 而人工鱼的数量越多, 就越容易寻找局部最优值, 这将有更多的人工鱼游到该位置, 这相当于蚁群算法信息素浓度增加, 类似于蚂蚁群算法中信息素正反馈的过程.

和蚁群算法不同的是, 人工鱼前进的方向是由当前状态最优的情况决定的, 并且在某些条件下会受到拥挤度的限制. 如果该地区人工与数量过多, 超过拥挤度的约束, 即使该地区有很高的价值, 人工鱼也不会聚集. 而在蚁群算法中, 信息素关联着蚁群个体之间的联系, 并且和启发式信息一起共同指导个体行为, 最终所有的蚂蚁都会聚集到同一条最佳路径上来[11].

两种算法搜索的方式不同, 从而个体的运动方式也各不相同, 这种差异也是由个体运动的目的不同造成的. 启发式信息在两种算法的优化中起到了作用, 因为个体的目的都是尽快寻找食物. 但是拥挤程度并完全不适用于蚁群算法, 因为蚁群寻找到食物后会搬回巢穴, 而鱼群在寻找到食物后会直接吃掉, 并不会存储, 而过多的人工鱼会导致其它人工鱼到这里之前, 食物已经被吃光, 所以拥挤度对算法的全局性起着关键的作用. 蚂蚁在寻找到食物并且搬回的过程中, 如果发现了更高效的路径, 则会转移到其他路径上去, 在选择最短路径上, 蚂蚁的数量并不受到拥挤度的限制, 选择的蚂蚁越多, 对结果越好, 但是如果这条路径并非最优选择, 算法可能会被局限在局部最优解中.

因此, 两种算法之间的核心区别是拥挤程度在优化过程中起着指导作用. 换句话说, 鱼群算法类似于于在蚁群算法中引入拥挤度的概念, 并且算法优化过程中的拥挤程度总是有着重要作用. 拥挤程度的引入可以避免蚁群算法早期, 蚂蚁过早地聚集到信息素高的路径上来, 避免算法的早熟, 提高算法的全局优化能力. 算法后期, 拥挤度的作用便会从正作用慢慢转向副作用, 对算法的收敛速度产生影响. 换句话说, 在早期优化中的拥挤程度可以提高算法的优越性能, 但是在后期优化中对性能会有一定的负面影响. 如何正确使用拥挤度这个因素, 是保证算法高效, 准确的关键[12].

3.2 先鱼群再蚁群的混合算法

为了保证较优的可行解, 产生初始信息素分布, 防止解过早集中到信息素较高的路径上来, 首先使用鱼群算法保证求解的宽度, 并设置更新迭代率, 当算法的更新率少于设定值时, 转入蚁群算法; 转入蚁群算法后, 更新信息素, 迭代寻找较优解, 当结果不再变化时, 再次转入鱼群算法, 并且调整更新率[13]. 算法步骤如下:

Step 1. 初始化模型和子空间.

Step 2. 设置 AFSA 的最大最小迭代次数 $I_{\max}$, $I_{\min}$, 更新率 R_{ratio} 和拥挤度. 生成 m 条人工鱼, 视野长度 V_d, 拥挤度 δ 和移动步长 D_{step}, 产生初始人工鱼群 $F(0)$.

Step 3. 每条人工鱼进行一次迭代, 迭代过程中分别执行觅食, 聚群, 追尾三种行为, 迭代结束后, 将结果与公告板相比较, 更新最优值.

Step 4. 在迭代次数的范围内, 将此次的迭代结果和上一次结果相比较, 计算更新率 R. 如果 $R<R_{\text{ratio}}$, 说明优化效果降低, 转入 Step 5, 如果 $R>R_{\text{ratio}}$, 说明鱼群算法依然优化效果良好, 转入 Step 3.

Step 5. 根据鱼群算法的迭代结果, 生成 m 只蚂蚁, 初始化信息素 $\tau_{i,j}(0)$.

$$\tau_{i,j}(0)=\frac{M_n}{\sum C^{mn}}$$

其中, M 代表算法中蚂蚁的总数量, n 是工件数目, $\sum C^{mn}$ 表示总完工时间.

Step 6. m 只蚂蚁随机放入工件上, 根据每条路径上的信息素浓度, 计算状态转移概率 $P_{i,j}^{k}(t)$ 并且由位置 i 转移到位置 j, 多次迭代后, 得到最优结果.

Step 7. 判断算法是否达到最大迭代次数后者满足输出条件, 若满足, 输出结果; 若不满足, 以蚁群算法的结果为新的人工鱼群 $F(1)$, 并且调整拥挤度阈值和更新率 R_{ratio}, 转入 Step 3.

3.3 直接嵌入拥挤度的蚁群算法

将拥挤度直接嵌入蚁群算法的迭代过程中, 在蚂蚁选择每条路径时, 首先判断路径上的拥挤度, 如果拥度小于阈值, 则继续选择该路径, 如果拥挤度大于阈值, 则随机选择可行的其它路径. 蚁群算法路径上的拥挤度为:

$$q_{i,j}=\frac{2\tau_{i,j}(t)}{\sum\limits_{i\neq j}\tau_{i,j}(t)} \tag{10}$$

算法步骤如下:

Step 1. 初始化时刻 t、信息素、拥挤度阈值 δ_t 和迭代次数限制 I_r.

Step 2. m 只蚂蚁随机放入工件上, 每只蚂蚁根据信息素选择路径, 状态转移概率 $P_{i,j}^{k}(t)$ 表示 t 时刻蚂蚁由位置 i 转移到位置 j 的概率, 如公式 (8).

Step 3. 每当蚂蚁选择一条路径, 通过公式 (10) 计算路径的拥挤度 $q_{i,j}$, 如果 $q_{i,j}>\delta_t$, 表示路径过于拥挤, 蚂蚁从可行领域内随机选择其它路径; 如果 $q_{i,j}<\delta_t$, 表示路径不太拥挤, 蚂蚁从 i 转移到 j.

Step 4. 利用 BF 分批规则对工件序列分批, 计算目标函数值; 更新信息素 τ_{ij}, 如公式 (9), 和拥挤度阈值 δ_t:

$$\delta(t)=1-e^{-ct} \tag{11}$$

其中, c 为阈值变化系数.

Step 5. 重复 Step 2、3 和 4, 直到所有蚂蚁都选择一条路径或者达到最大迭代次数.

两种算法的流程图如图 1 所示.

4 仿真实验与分析

4.1 实验设计

为了验证本文提出的算法有效性, 本文的实验需要兼顾三个方面要素: 工件尺寸的变化, 工件数的变化, 工件加工时间的变化. 如表 1 所示, 工件尺寸的大小和加工时间我们采用离散型随机分布的方式, 随机决定工件尺寸的大小; 工件数量方面, 我们通过数量递增的对比结果来对比算法的有效性; 算法的有效性我们考虑算法的加工时间和批利用率、负载率的均值.

各个参数的设置为: 蚂蚁数量 m 为 20、Q=100、$\tau_{i,j}$=1/n、ρ=0.5、α=1、β=1.

4.2 实验结果与分析

加工时间和工件数都是离散的, 为了保证实验的有效性, 我们根据工件的数量分为三类, 分别对结果加以对比, 分别如表 2 和表 3 所示.

在表 2, 表 3 中可以看出, 在算法寻优的过程中, 蚁群算法的性能要优于鱼群算法, 但是蚁群算法本身的早熟性, 导致寻优结果局部最优, 但是将蚁群算法和鱼群算法相结合, 利用鱼群算法中拥挤度因子, 并与蚁群算法相结合, 可以有效地避免早熟, 并且对于寻找最优解, 减少寻优时间有着一定的帮助. 通过混合算法 3.1 和 3.2 的比较, 混合算法 3.2 对于工件数量较小, 迭代次数较少的问题有较高效率. 而混合算法 3.1 对于工件数量较多, 迭代次数较多的算法有较高的性能.

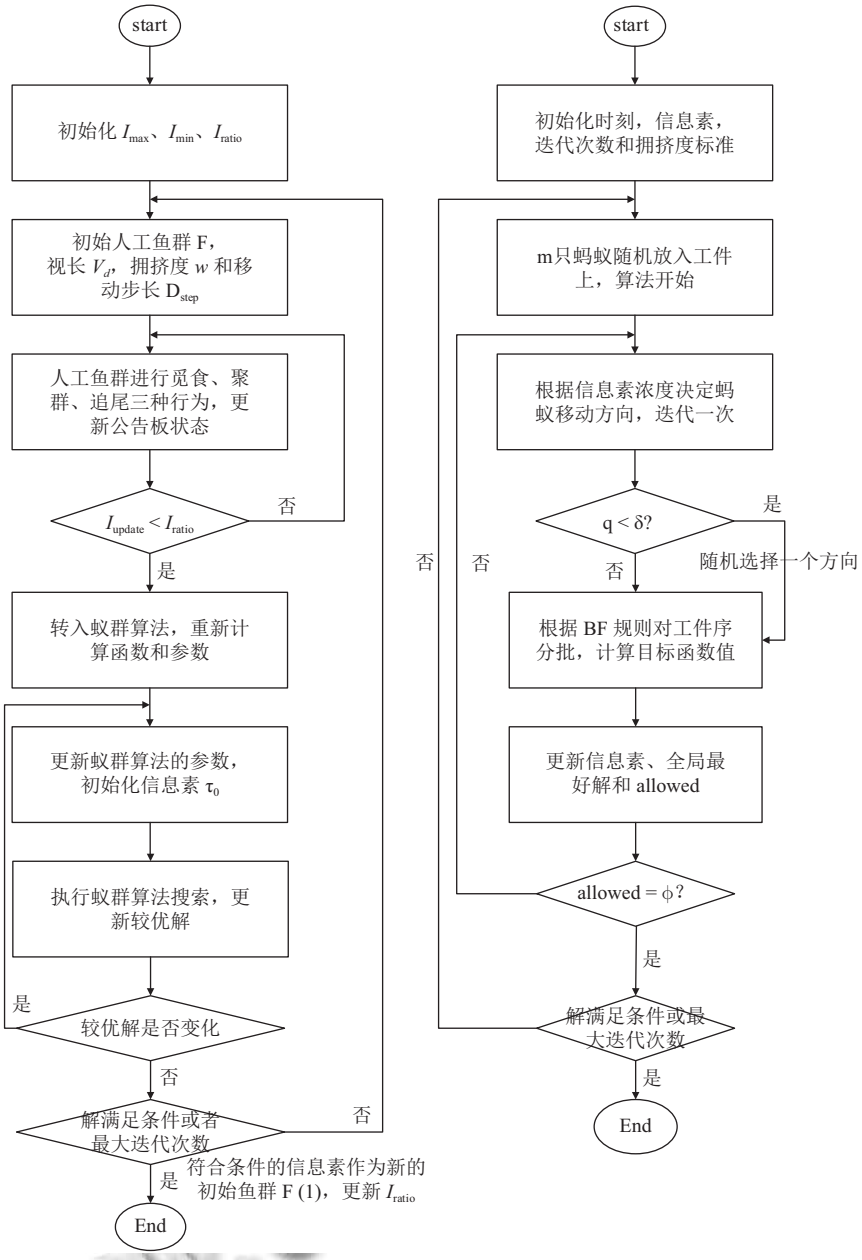

图 1 两种算法流程图

表 1 实验中的参数设置

要素	等级和取值
工件数	J1类: 20 J2类: 50 J3类: 100
工件加工时间	服从[1, 10]离散均匀分布
工件尺寸	服从[1, 10]离散均匀分布

5 总结

本文针对差异工件批调度问题, 并考虑到蚁群算法的早熟性问题, 将鱼群算法与之相结合, 利用鱼群算法中鱼群防止过度聚集, 拥挤度限制的方法, 避开蚁群算法在寻优前期早熟死循环. 对于算法结果, 本文通过负载率和利用率均值的相比较来判断算法结果的优劣, 迭代次数的比较来对比算法效率. 通过对比的结果发现, 混合算法前期可以有效避免早熟, 而且在算法后期又可以加快收敛, 从而使寻优个体能够加快寻找到满意解.

但是对于混合算法中, 混合算法 3.1 的更新率、拥挤的阈值, 以及混合算法 3.2 中迭代次数限制, 都需要

人为确定, 但是针对不同情况下, 不同的值对结果有着不同的影响, 那么工件数量、工件尺寸的分布、工件的加工时间和机器的尺寸和这些值的关系还需要进一步的探究.

表 2 ACO 和 AFSA 算法对比

	ACO		AFSA	
	平均迭代次数	利用率和负载率均值	平均迭代次数	利用率和负载率均值
20	296	0.402	302	0.412
50	958	0.474	1204	0.518
100	2121	0.648	2956	0.894

表 3 混合算法 3.1 和 3.2 对比

	混合算法3.1		混合算法3.2	
	平均迭代次数	利用率和负载率均值	平均迭代次数	利用率和负载率均值
20	266	0.380	254	0.382
50	766	0.394	759	0.364
100	1356	0.488	1526	0.504

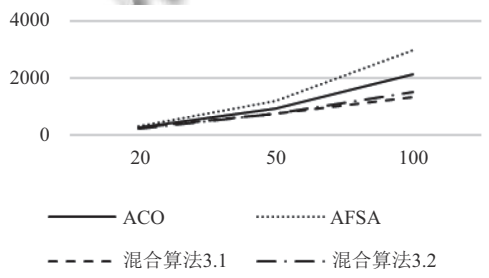

图 2 算法迭代次数对比

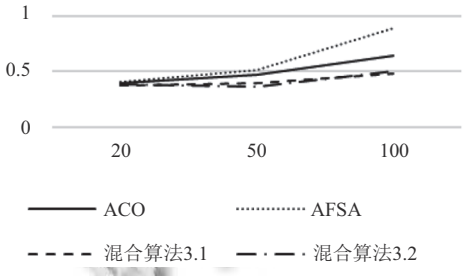

图 3 利用率负载率均值对比

参考文献

1 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. 管理科学学报, 2009, 12(6): 72–82.

2 Ghazvini FJ, Dupont L. Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes. International Journal of Production Economics, 1998, 55(3): 273–280. [doi: 10.1016/S0925-5273(98)00067-X]

3 Melouk S, Damodaran P, Chang PY. Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. International Journal of Production Economics, 2004, 87(2): 141–147. [doi: 10.1016/S0925-5273(03)00092-6]

4 Damodaran P, Manjeshwar PK, Srihari K. Minimizing makespan on a batch-processing machine with non-identical job sizes using genetic algorithms. International Journal of Production Economics, 2006, 103(2): 882–891. [doi: 10.1016/j.ijpe.2006.02.010]

5 王笑蓉, 吴铁军. Flowshop 问题的蚁群优化调度方法. 系统工程理论与实践, 2003, 23(5): 65–71.

6 姜桦, 李莉, 乔非, 等. 蚁群算法在生产调度中的应用. 计算机工程, 2005, 31(5): 76–78, 101.

7 王常青, 操云甫, 戴国忠. 用双向收敛蚁群算法解作业车间调度问题. 计算机集成制造系统, 2004, 10(7): 820–824.

8 卢辉斌, 范庆辉, 贾兴伟. 一种改进的自适应蚁群算法. 计算机工程与设计, 2005, 26(11): 3065–3066. [doi: 10.3969/j.issn.1000-7024.2005.11.064]

9 李晓磊, 邵之江, 钱积新. 一种基于动物自治体的寻优模式: 鱼群算法. 系统工程理论与实践, 2002, 22(11): 32–38. [doi: 10.3321/j.issn:1000-6788.2002.11.007]

10 李晓磊, 路飞, 田国会, 等. 组合优化问题的人工鱼群算法应用. 山东大学学报 (工学版), 2004, 34(5): 64–67.

11 侯景伟, 孔云峰, 孙九林. 基于多目标鱼群-蚁群算法的水资源优化配置. 资源科学, 2011, 33(12): 2255–2261.

12 王联国, 洪毅, 赵付青, 等. 一种改进的人工鱼群算法. 计算机工程, 2008, 34(19): 192–194. [doi: 10.3969/j.issn.1000-3428.2008.19.065]

13 修春波, 张雨虹. 基于蚁群与鱼群的混合优化算法. 计算机工程, 2008, 34(14): 206–207, 218. [doi: 10.3969/j.issn.1000-3428.2008.14.073]