运筹学与控制论

DOI:10.11721/cqnuj20180518

平行机最小化误工损失调度问题的粒子群算法*

陈 鑫[1], 李昕昀[1], 谢鹏宇[2]
(1. 辽宁工业大学 电子与信息工程学院, 辽宁 锦州 121001;
2. 罗格斯大学 新布朗斯维克研究生院, 新泽西 新布朗斯维克 08901-1178)

摘要:【目的】讨论非同类机环境下最小化任务总误工损失的调度问题。任务的误工损失是与交付期有关的一种惩罚量,该惩罚量的值等于任务滞后于交付期加工的部分。【方法】设计了一个粒子群算法求解该问题,并以数值实验进行验证。【结果】针对问题特性,对粒子群算法中的粒子表达方式、运算操作、初始解生成、种群更新方法等进行了重新定义。【结论】数值实验表明,算法处理该问题时可获得性能良好的解,并且运行时间也在可接受范围之内。

关键词:误工损失;非同类机;截止期;粒子群算法
中图分类号:TP18 **文献标志码:**A **文章编号:**1672-6693(2018)05-0010-07

调度问题是组合优化、管理科学、计算机科学等领域的经典问题之一,主要研究在满足一定约束的前提下将一定时间内的任务(Job)分配给有限的处理机(Machine),以优化一个或多个目标[1]。实际生产的多样性,导致调度问题的优化目标也不尽相同;其中,任务的误工损失(Late work)是一个与交付期(Due date)有关的惩罚量,误工损失的值等于任务滞后于交付期加工的部分;而最小化任务总误工损失的调度模型,由于能够很好地刻画某些生产环境的场景,引起了诸多学者的兴趣。

任务误工损失这一概念最早由 Blazewicz 于 1984 年提出[2],他的灵感来自于控制系统中的信息收集,因此将之称为"Information loss",在三参数表示法中用 Y 来表示;1992 年,Potts 和 Van Wassenhove[3] 研究了同一函数,将之命名为"Late work",并用 V 来表示。由于这两篇文献研究的是相同的目标,为统一起见,后期的学者大多使用"Late work"这一术语,用 Y 表示。

从 1984 年问题提出至今,对最小化任务总误工损失调度问题的研究涵盖了单机(Single machine)、平行机(Parallel machines)、专用机(Dedicated machines)等多种处理机环境。对于单机环境,Potts 和 Van Wassenhove 于 1992 年证明了问题 $1||Y$ 是 NP 难的,并给出了一个伪多项式时间的动态规划算法[3];2000 年,Woeginger 给出了问题 $1||Y$ 的完全多项式时间近似方案(Fully polynomial time approximation scheme, FPTAS)[4];2005 年,Lin 和 Hsu 研究了带释放时间和允许任务重启的调度问题,即 $1|r_j, \text{pmtn}|Y$,并给出了时间复杂度为 $O(n\log n)$ 的最优算法[5];Yin 等人给出了单机带维护时间($1|\text{MA}|Y$)问题的多项式近似方案[6];马露和张新功研究了双代理的单机调度问题,并给出拟多项式时间的动态规划算法[7]。

对于专用机环境,Blazewicz 等人分别给出问题 $O_2|d_j=d|Y_w$[8], $F_2|d_j=d|Y_w$[9] 和 $J_2|d_j=d, n_j\leqslant 2|Y_w$[10] 的复杂性证明,并设计了相应的伪多项式时间算法;Lin 等人研究了 $F_2||Y$ 问题,给出了一个分支定界算法[11];Pesch 和 Sterna 将问题扩展至带释放时间的多流水机环境(即 $F|r_j|Y$),设计了一个遗传算法并通过数值实验对算法的有效性进行验证[12]。

误工损失这一目标函数最早用于研究平行机环境。Blazewicz 等人首先研究了 $P||Y, P|r_j, \text{pmtn}|Y_w$ 和 $Q|r_j, \text{pmtn}|Y_w$ 等问题[2,13];最近,Abasian 等人研究了平行机带通讯延迟的模型,即 $P_m|\text{prec}, \text{comu}|Y_w$[14];Chen 等人证明了问题 $P_m|d_j=d|Y$ 是 NP 难的,并且给出了 $P_2|d_j=d|Y$ 的一个伪多项式时间动态规划算法[15]。

* **收稿日期:**2018-02-09 **修回日期:**2018-09-10 **网络出版时间:**2018-09-26 13:25
资助项目:辽宁省博士科研启动基金(No. 20170520306)
第一作者简介:陈鑫,男,副教授,博士,研究方向为组合优化问题、调度问题的算法分析与设计,E-mail:chenxin.lut@hotmail.com
网络出版地址:http://kns.cnki.net/kcms/detail/50.1165.N.20180926.1325.014.html

由于 $P_m|d_j=d|Y$ 是 NP 困难问题，它的通用情况 $R_m||Y$ 无疑也是 NP 难的（其中 R_m 表示非同类平行机环境）。对于 NP 困难问题，面对大规模算例时很难存在有效时间内的精确算法，因此可以通过某些智能算法获得质量可接受的可行解[16-17]。基于上述思路，本文设计了一个求解 $R_m||Y$ 问题的粒子群算法，首先基于目标函数对粒子表示方法进行定义，然后重载了粒子群算法中的加、减、乘法操作，通过若干启发式规则生成初始解并进行种群更新，最终通过公共测试集对算法的性能进行测验。数值实验表明，无论从解的性能方面、还是运算时间方面，本文的粒子群算法都是求解问题 $R_m||Y$ 的一个有效算法。

1 问题描述

如前所述，任务的误工损失表示在其交付期之后所处理部分对应的惩罚，如图 1 所示[15]。为了表示方便，图 1 中假设所有任务的交付期均为 d，即 $d_j=d$。其中，任务 J_i 在 d 之前加工完成，因此误工损失 $Y_i=0$；任务 J_k 在 d 之前开始加工，但是在 d 之后才完成加工，则此时的误工损失为交付期之后加工的部分，即 $Y_k=C_k-d$，其中 C_k 为 J_k 的加工完成时间；任务 J_r 在 d 之后开始加工，这是一个完全误工的任务，因此误工损失 Y_r 为加工时间，即 $Y_r=p_r$。可见，任务 J_j 的误工损失 Y_j 表示为 $Y_j=\min\{p_j,\max\{0,C_j-d_j\}\}$。

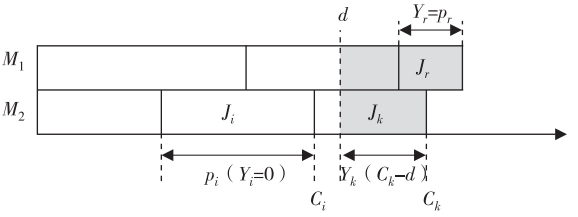

图 1　误工损失

Fig. 1　Illustration of late work parameter

因此，本文所研究的问题可描述如下。

输入：有 m 台非同类平行机和 n 个任务，其中任务 J_j 在处理机 M_i 上对应的加工时间为 p_{ij}（$1\leqslant i\leqslant m$，$1\leqslant j\leqslant n$）；每个任务都有对应的交付期 d_j。调度为非抢占模式，即一旦任务 J_j 在处理机 M_i 上开始加工，直至加工完毕这段时间内，M_i 只能处理 J_j，该处理不允许被中断。

输出：将 n 个任务分配到 m 台处理机上，目标为最小化所有任务的误工损失之和（$\sum Y_j$）。用三参数表示法，本文所研究的问题可表示为 $R_m||Y$。

2 求解 $R_m||Y$ 问题的粒子群算法

2.1 粒子群算法概述

粒子群算法是一种基于种群智慧的元启发式算法[16]，由 Kennedy 和 Eberhart 于 1995 年受自然界生物群体如鸟群、鱼群觅食过程的启发而提出[18]，并已广泛地应用于求解任务调度等优化问题[19-22]。

在基础结构中，粒子群包含 N 个在 D 维搜索空间自由移动的粒子，每个粒子是问题的一个可行候选解。粒子 Q_k 拥有自己的位置 x_k（表达一种可行的求解方案，用来评估该方案的质量）和速度 v_k（用来计算粒子下一次的移动方向及距离），算法的优化过程就是利用这些粒子之间的相互作用寻求全局最优解：粒子 Q_k 通过自身访问过的最佳位置（记作 p_{best_k}）和整个种群访问过的最佳位置（记作 g_{best}）来调整下一轮位置，以靠近最优解。在第 $(t+1)$ 次迭代过程中，粒子 Q_k 的速度和位置计算方法为：

$$v_k^{t+1}=wv_k^t+c_1r_1\times(p_{\text{best}_k}^t-x_k^t)+c_2r_2\times(g_{\text{best}}^t-x_k^t),\tag{1}$$

$$x_k^{t+1}=x_k^t+v_k^{t+1}。\tag{2}$$

其中，w 为惯性因子，反映了粒子原始速度对下一次速度的影响；常数 c_1，c_2 为学习因子，反映了粒子受到自身最优解和种群最优解吸引的程度；r_1，r_2 是在[0,1]范围内的随机变量。上标 t 为迭代指数，$p_{\text{best}_k}^t$ 表示粒子 Q_k 在前 t 轮迭代过程中找到的自身最优解，而 g_{best}^t 则表示所有粒子在前 t 轮迭代过程中找到的种群最优解。

2.2 粒子的表示方法

使用粒子群算法求解 $R_m||Y$ 问题，首先需要根据问题特性对粒子的位置、速度等数据结构进行定义。

2.2.1 *粒子位置的编码方案*　如前一节所述，粒子的位置表达一种可行的调度方案。文献[20]用一个 n 元整型数组表示一种方案，数组中第 j 个元素的值即为任务 J_j 所分配的处理机编号。但该文献所研究的问题是最小化制造周期问题，无需关注在每台处理机上任务之间的先后顺序。本文研究的非同类机最小化误工损失问题，不仅需要关注任务的处理机分配，还需要考虑每一台处理机上任务的相对位置。因此，本文采用文献[23-24]中的方式对粒子位置进行编码：用 n 元实数型数组表示调度方案，其中第 j 个元素整数部分的值表示任务 J_j 所分配

的处理机编号,小数部分的值用来计算此任务在该处理机上的位置(按小数部分值的非降序排列)。

例如,系统中有2台处理机、5个任务,某粒子位置为 $x_k=[1.287,2.542,1.439,2.116,2.123]$,则对应的调度方案如图2所示。

2.2.2 *粒子速度的编码方案* 速度在物理中的定义包括方向和大小两个信息,决定了物体移动的朝向和距离。而从(1),(2)式可以看出,粒子群算法中的速度 v 主要是与位置 x 做加法运算,以在迭代中改变粒子的位置。因此,本文用与粒子位置相同的数据结构表示粒子速度;而为了控制位置的变化幅度,速度数组中引入"缺省值" f 的概念,即当数组内的某一元素为缺省值时,该元素对应的位置信息在下一轮迭代中不发生改变。

例如,若 t 时刻粒子 k 的位置 $x_k^t=[d_1,d_2,d_3,d_4,d_5]$,由(1)式得到的 $t+1$ 时刻速度为 $v_k^{t+1}=[f,d_4,d_5,f,f]$,则 $t+1$ 时刻粒子 k 位置的计算方法为:第1,4,5位保留 x_k^t 的值(因为速度数组第1,4,5位为缺省值 f);而第2,3位的值按照下一节中重载的加法运算符进行运算。

2.2.3 *随机变量 r_1,r_2 的表达方式* $R_m||Y$ 问题中的粒子位置、速度都是 n 元数组,根据(1)式的含义,无法将 r_1,r_2 定义为一个简单的随机数参与乘法运算。因此,本文将随机变量 r_1,r_2 同样定义为一个 n 元数组,每一位的值为0或者1。若数组中某一位值为0,表示本位置的全局/局部最优解不参与合成粒子速度,最终得到新的粒子速度在该位置为缺省值 f;若该位值为1,则表示本位置的全局/局部最优解参与合成粒子速度。具体计算方法见下面关于乘法运算符的重载方案。

2.3 运算符重载

为 $R_m||Y$ 问题构造粒子群算法的另一项主要工作是根据问题特性对(1),(2)式中的运算符进行重载,重载方案既要保证搜索的可靠性、还要保证解的有效性,即:新的位置不会偏离原位置太远,而且要向当前种群最优解和自身最优解靠近;同时,新的位置依然能够表达一种可行的调度方案。本文的运算符重载方案参考文献[20]设计。

2.3.1 *减法运算符重载* 根据(1)式可以看出,减法运算主要表达了当前位置与最优位置的"差距",以全局/局部最优位置为被减数、当前位置为减数,得到的结果将影响粒子朝向全局/局部最优位置的分速度。本文定义的减法操作 $C=A-B$ 为:对于 A 和 B 的每一个位置 i,若 A 和 B 上该位置的值相同,则结果的对应位置的值为缺省值 f;若 A 和 B 在该位置上的值不同,则结果对应位置的值取 A(较优解)中的值,如图3所示。

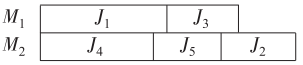

图2 粒子所表达的调度方案

Fig. 2 The schedule of the particle

A	d_1	d_2	d_3	d_4	d_5
B	d_1	d_2	d_3	d_4	d_5
$C=A-B$	f	d_2	d_3	d_4	f

图3 减法操作示例

Fig. 3 Illustration of subtraction

2.3.2 *乘法运算符重载* 乘法操作用于随机变量与减法差值之间做运算,可以理解为对位置差距上分配的"权重"。本文重载乘法运算符 $C=A\times B$ 的方案为:对于 A 和 B 上的每一个位置 i,若 $A_i=0$,则 $C_i=f$;若 $A_i=1$,则 $C_i=B_i$,如图4所示。注意,运算数 A 为随机变量 r_1 或 r_2,每一位的值非0即1。

2.3.3 *加法运算符重载* 加法运算用来合成(1),(2)式中新的速度和位置,是粒子群算法中最重要的操作。一方面,加法运算要保证合成的可靠性(新的位置根据自身当前位置、自身搜索得到的最佳位置、种群搜索得到的最佳位置合成得出);另一方面,还需要保证操作的合理性,即新的位置表达的也是一种可行调度。本文定义加法操作 $C=A+B$ 如下。

步骤1 令 $C=A$;

步骤2 从[1,…,n]中按照概率 P 选择出一些位置坐标,构成集合 Q;

步骤3 选择 Q 中的一个坐标 i;

步骤3.1 若 B_i 是缺省值 f,转到步骤4;

步骤3.2 若 $B_i=C_i$,转到步骤4;

步骤3.3 在 C 中寻找值为 B_i 的坐标 j,交换 C_i 和 C_j 的值;

步骤4 在 Q 中移除 i。如果 Q 中还有元素,转回步骤3;否则终止。

可见,加法过程可以看作 A 模仿 B 进行部分改动的过程,而两两交换保证了在改动过程中 C 的合理性不会被破坏。因此,当 A 是一个可行解的时候,C 最终也是一个可行解。加法运算的示例如图5所示。

A	1	0	1	1	0
B	f	d_2	d_3	d_4	f
$C=A\times B$	f	f	d_3	d_4	f

图 4　乘法操作示例

Fig. 4　Illustration of multiplication

A	d_2	d_1	d_5	d_3	d_4
B	f	f	d_3	d_4	f
$C=A+B$	d_2	d_1	d_3	d_5	d_4

图 5　加法操作示例

Fig. 5　Illustration of addition

图 5 中,先将 C 初始化为 A,假设随机选中的坐标集合 Q 为{2,3}。因 B_2 是缺省值 f,所以该坐标被跳过;对于坐标 3,根据步骤 3.3,在 C 中找到值为 B_3 的元素(C_4),交换 C_3 和 C_4,最终得到数组 $C=[d_2,d_1,d_3,d_5,d_4]$。

2.4 算法流程

1) 种群初始化。本文构造的种群大小为 $N=50$,即粒子的数量为 50。其中 4 个粒子采用如下的启发式规则进行初始化。

RSPT-LPT **规则**　将每个任务分配到处理所花费时间最小的处理机上,每台机器上的任务按处理时间非递增排列;

RSPT-EDD **规则**　将每个任务分配到处理所花费时间最小的处理机上,每台机器上的任务按交付期非递减排列;

min $C_{\max}$ **规则**　迭代 n 次构造一个可行调度,每次将一个任务分配至合适的处理机使得当前系统总的制造周期最小;

min $Y_{\max}$ **规则**　迭代 n 次构造一个可行调度,每次将一个任务分配至合适的处理机使得当前系统总的误工损失最小。

而对于剩余的 46 个粒子,采用随机的方法构造其初始解。

2) 算法终止条件。算法在满足终止条件时将会停止迭代,输出当前搜索到的全局最优解。本文使用了两个终止指标。① 最大迭代次数:当迭代次数超过最大迭代次数时,算法终止。本文将最大迭代次数设置为 10 000;② 最大未改变迭代次数:当最优解未更新的迭代次数超过所能容忍的最大迭代次数时,算法终止。本文将之设为 1 000。当这两个终止条件满足任意一个时,算法终止。

3) 算法整体框架。综上,本文粒子群算法的整体流程可以表述如下:

Initialize initial population;　//种群初始化

while terminated condition is not met

　Update each particle by (1) and (2);　//使用重载后的运算符进行种群更新

Output *gBest*;　//输出搜索到的最优解

3 实验及结果分析

算法使用 C++语言编程,开发平台为 Visual Studio 2017,测试用的机器为 Intel® Core™ i7-4700 2.40 GHz,8 Gb RAM 的笔记本电脑。

3.1 实验数据

本文使用的实验数据参照文献[11]的生成方法,并根据问题特点进行了一些改进。任务 J_j 在处理机 M_i 上的处理时间 $p_{ij}\sim U(5,50)$;而对于交付期的生成方法,首先计算任务的平均处理时间 $\overline{p_j}=\frac{1}{m}\sum_{i=1}^{m}p_{ij}$,并按照$\overline{p_j}$的非递减顺序对任务进行重新排序。新序列中第 k'个任务的交付期 $d_{k'}\sim U\left(\overline{p_{k'}},\overline{p_{k'}}+\frac{1}{\beta}\sum_{i=1}^{k'}\overline{p_{k'-i+1}}\right)$,其中 β 为松紧系数,通过控制 β 的值可以得到交付期松紧程度不同的实验数据。

3.2 结果分析

本文采用文献[12]中的方法对粒子群算法的求解质量进行分析,即通过比较粒子群算法对于几种启发式算法提升率的值$\frac{Y_A-Y_{PSO}}{Y_A}$来评估算法的求解质量,其中 Y_A,Y_{PSO}分别指对于某一算例而言,算法 A 和本文粒子群算法得到的误工损失值;为了模拟多种任务交付期的情况,分别生成了 $\beta=20,30,40$ 时的任务数据;考虑到任务数量 n 和机器数量 m 也将影响算法的效率,使用了不同的(m,n)组合进行实验,其中 m 分别取 3,5,10 和 20,而

对于每一个 m 值,n 的取值分别为 $3m$,$5m$,$7m$ 和 $10m$。针对每一组参数设置,分别运行 50 次实验取平均值,共计实验 2 400 组。

表 1 (m,n)组合不同时粒子群算法的性能表现

Tab.1 The performance of PSO for different settings of (m,n)

m	n	CPU 处理时间/s	RSPT-LPT	RSPT-EDD	min C_{max}	min Y_{max}
3	9	0.66	86.37	49.57	64.22	63.99
	15	0.81	81.20	31.93	54.73	55.77
	21	1.07	78.97	25.08	52.49	53.26
	30	1.45	77.14	18.68	51.22	51.73
	组内平均值	1.00	80.92	31.31	55.66	56.19
5	15	1.00	91.32	64.40	76.77	77.11
	25	1.49	86.00	42.57	68.36	68.30
	35	2.55	83.62	31.69	65.47	64.76
	50	3.82	81.49	22.50	61.80	60.93
	组内平均值	2.21	85.61	40.29	68.10	67.77
10	30	2.75	95.63	79.18	89.16	89.13
	50	6.20	89.80	50.51	78.13	77.33
	70	8.27	86.50	32.47	71.32	72.05
	100	7.14	82.62	12.65	64.49	64.04
	组内平均值	6.09	88.64	43.70	75.77	75.64
20	60	8.89	97.28	83.42	93.49	93.57
	100	8.40	88.76	30.53	75.24	74.96
	140	7.90	85.76	11.74	70.31	70.10
	200	6.32	84.56	2.85	67.93	68.17
	组内平均值	7.88	89.09	32.14	76.74	76.70
平均值		4.30	86.06	36.86	69.07	69.07

表 1 分析了在不同(m,n)组合下,粒子群搜索算法对 RSPT-LPT,RSPT-EDD,min C_{max},min Y_{max} 这 4 种简易启发式算法的提升率。例如,当 $m=3$,$n=9$ 时,粒子群算法对于 RSPT-EDD 的提升率为 49.57%,而对于 min Y_{max} 的提升率为 63.99%,此组算例的平均运行时间为 0.66 s。观察可以发现,粒子群优化算法在不同(m,n)组合的情况下提升率的变化情况为:随着机器数目的增加,算法的综合提升率逐渐增加;随着任务数/机器数比的逐渐增加,算法的综合提升率逐渐下降。

表 2 反映了粒子群算法在松紧系数 β 不同时的表现:随着 β 的增加,算法的提升率逐渐下降。即随着任务的交付期越来越紧,粒子群算法对 4 种简易启发式算法的综合提升率将会逐渐下降。

表 2 β不同时粒子群算法的性能表现

Tab.2 The performance of PSO for different β

β	RSPT-LPT	RSPT-EDD	min C_{max}	min Y_{max}
20	87.84	44.76	71.84	71.62
30	85.69	35.01	68.32	68.44
40	84.67	30.82	67.05	67.16
平均值	86.06	36.86	69.07	69.07

4 结论

本文研究了非同类机环境下的最小化任务误工损失的调度问题,并设计了一个粒子群算法求解问题。针对问题特点,首先构造了解的表达方式,并对原始粒子群算法中的操作符进行重新定义,结合初始解生成、算法终

止条件等策略完成算法设计。实验过程中，通过对比不同规模和不同松紧程度交付期的算例，验证了算法求解问题的性能，且该算法的运行时间也在可接受范围之内。

本文研究的问题目前尚无有效的精确算法，因此，未来的研究工作可以关注精确算法的设计，以求解中小规模的算例；同时，还可以考虑设计其他的元启发式算法如禁忌搜索、遗传算法等，与本文的粒子群算法进行比较，以找到更适合解决该问题的智能算法。

参考文献：

[1] PINEDO M. Scheduling: theory, algorithms, and systems [M]. 4th Edition. New York: Springer, 2012.

[2] BLAZEWICZ J. Scheduling preemptible tasks on parallel processors with information loss[J]. Technique et Science Informatiques, 1984, 3(6): 415-420.

[3] POTTS C N, Van WASSENHOVE L N. Single machine scheduling to minimize total late work[J]. Operations Research, 1992, 40(3): 586-595.

[4] WOEGINGER G J. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? [J]. Informs Journal on Computing, 2000, 12(1): 57-74.

[5] LIN B M, HSU S. Minimizing total late work on a single machine with release and due dates[C]//SIAM Conference on Computational Science and Engineering (CSE05), February 11－13, 2005, Orlando. Philadelphia: SIAM, 2005.

[6] YIN Y, XU J, CHENG T C E, et al. Approximation schemes for single-machine scheduling with a fixed maintenance activity to minimize the total amount of late work [J]. Naval Research Logistics, 2016, 63(2): 172-183.

[7] 马露，张新功. 关于总误工损失的两个代理单机排序问题 [J]. 运筹学学报，2017，21(1)：13-22.
MA L, ZHANG X G. Two-agent scheduling problem about total late work on a single machine[J]. Operations Research Transactions, 2017, 21(1): 13-22.

[8] BLAZEWICZ J, PESCH E, STERNA M, et al. Open shop scheduling problems with late work criteria[J]. Discrete Applied Mathematics, 2004, 134(1): 1-24.

[9] BLAZEWICZ J, PESCH E, STERNA M, et al. The two-machine flowshop problem with weighted late work criterion and common due date[J]. European Journal of Operational Research, 2005, 165(2): 408-415.

[10] BLAZEWICZ J, PESCH E, STERNA M, et al. A note on the two machine job shop with the weighted late work criterion[J]. Journal of Scheduling, 2007, 10(2): 87-95.

[11] LIN B M, LIN F, LEE R. Two-machine flow-shop scheduling to minimize total late work[J]. Engineering Optimization, 2006, 38(4): 501-509.

[12] PESCH E, STERNA M. Late work minimization in flow shops by a genetic algorithm[J]. Computers & Industrial Engineering, 2009, 57(4): 1202-1209.

[13] BLAZEWICZ J, FINKE G. Minimizing mean weighted execution time loss on identical and uniform processors[J]. Information Processing Letters, 1987, 24(4): 259-263.

[14] ABASIAN F, RANJBAR M, SALARI M, et al. Minimizing the total weighted late work in scheduling of identical parallel processors with communication delays[J]. Applied Mathematical Modelling, 2014, 38(15): 3975-3986.

[15] CHEN X, STERNA M, HAN X, et al. Scheduling on parallel identical machines with late work criterion: Offline and online cases[J]. Journal of Scheduling, 2016, 19(6): 729-736.

[16] TALBI E G. Metaheuristics: from design to implementation[M]. New Jersey: John Wiley & Sons, 2009.

[17] 陈磊，霍永亮，霍波陶. 基于混合遗传算法的物流车辆调度优化[J]. 重庆师范大学学报(自然科学版)，2015，32(2)：7-12.
CHEN L, HUO Y L, HUO B T. Vehicle schedule optimization of logistics based on combinational genetic algorithm[J]. Journal of Chongqing Normal University (Natural Science), 2015, 32(2): 7-12.

[18] KENNEDY J, EBERHART R. Particle swarm optimization[C]//Proceedings of IEEE International Conference on Neural Networks. Perth, Australia: IEEE, 1995, 4: 1942-1948.

[19] 张长胜，孙吉贵，欧阳丹彤，等. 求解车间调度问题的自适应混合粒子群算法[J]. 计算机学报，2009，32(11)：2137-2146.
ZHANG C S, SUN J G, OU-YANG D T, et al. A self-adaptive hybrid particle swarm optimization algorithm for flow shop scheduling problem[J]. Chinese Journal of Computers, 2009, 32(11): 2137-2146.

[20] KASHAN A H, KARIMI B. A discrete particle swarm optimization algorithm for scheduling parallel machines [J]. Computers & Industrial Engineering, 2009, 56(1): 216-223.

[21] 张其亮，陈永生，韩斌. 改进的粒子群算法求解置换流水车间调度问题[J]. 计算机应用，2012，32(4)：1022-1024.
ZHANG Q L, CHEN Y S, HAN B. Improved particle swarm optimization for permutation flowshop scheduling problem[J]. Journal of Computer Applications, 2012, 32(4): 1022-1024.

[22] 宫华，张彪，许可，等. 基于粒子群算法的带有运输衔接的应急物资运输路径优化问题[J]. 重庆师范大学学报(自然

科学版),2015,32(3):23-29.
GONG H,ZHANG B,XU K,et al. A transportation routing problem of emergency materials based on particle swarm optimization algorithm[J]. Journal of Chongqing Normal University (Natural Science),2015,32(3):23-29.
[23] NIU Q,ZHOU T,WANG L. A hybrid particle swarm optimization for parallel machine total tardiness scheduling [J]. International Journal of Advanced Manufacturing Technology,2010,49:723-739.
[24] FANG K T,LIN B M T. Parallel-machine scheduling to minimize tardiness penalty and power cost[J]. Computers & Industrial Engineering,2013,64(1):224-234.

Operations Research and Cybernetics

A Particle Swarm Optimization Algorithm for Scheduling on Unrelated Machines with Late Work Criterion

CHEN Xin[1], LI Xinyun[1], XIE Pengyu[2]
(1. School of Electronics and Information Engineering, Liaoning University of Technology, Jinzhou Liaoning 121001, China;
2. New Brunswick Graduate School, State University of New Jersey, New Brunswick, New Jersey 08901-1178, USA)

Abstract: [**Purposes**]Studying scheduling problem on unrelated machines with late work criterion. [**Methods**]A particle swarm optimization (PSO) algorithm is designed to solve it. [**Findings**]A job's late work is a kind of due date related punishment criterion, which equals to the late part of this job, i.e. the part processed after its due date. Based on the problem characteristics, the data-structures, operations, solution initialization and iteration methods are redefined to develop the particle swarm optimization algorithm. [**Conclusions**] Numerical experiments show that it is a reasonable solution, from the views of algorithm-performance, as well as time-consumption.

Keywords: late work criterion; unrelated machines; due date; particle swarm optimization

(责任编辑 黄 颖)