文章编号: 1001-0920(2008)10-1092-06

带运输时间的无等待供应链在线调度问题研究

常桂娟[1,2], 张纪会[1]
(1. 青岛大学 复杂性科学研究所, 山东 青岛 266071; 2. 青岛农业大学 理学院, 山东 青岛 266109)

摘 要: 研究了供应链在线调度问题. 该问题具有工件无等待、工序之间存在运输时间、加工时间介于一个区间等特点,制造商随时可能接到顾客订单,订单到达前,所有信息如订单数量、到达时间及加工时间等均未知. 研究了在不改变已有工件调度的情况下,使用资源的可用时间区间最早完成临时订单的算法. 计算机仿真表明,使用该算法求解大规模临时订单问题是十分有效的.

关键词: 在线调度; 供应链; 微粒群; 实时
中图分类号: F273 **文献标识码**: A

No-wait on-line supply chain scheduling problems with transfer time

CHANG Gui-juan[1,2], *ZHANG Ji-hui*[1]
(1. Institute of Complexity Science, Qingdao University, Qingdao 266071, China; 2. College of Science, University of Qingdao Agricultural, Qingdao 266109, China. Correspondent: CHANG Gui-juan, E-mail: lucycgj@163.com)

Abstract: This paper considers on-line supply chain scheduling problems which is characterized by no-wait, transfer time between operations, processing time between two positive limits, etc. Manufacturer may receive orders from customers at any moment. There is no any information about the coming orders such as the orders quantity, release and processing time, etc. The paper gives an algorithm which can complete the orders within the time confined by available resources without changing other jobs' schedule. The simulation results show the effectiveness of the algorithm for solving large-scale order.

Key words: On-line scheduling; Supply chain; Particle swarm optimization; Real-time

1 引 言

供应链是一个全球化的组织网络,通过这些组织间的合作能以最低的成本和最快的速度提高供应商和消费者之间的原料流和信息流. 在这样一个系统里,调度目标不再是最优调度一系列的任务,而是系统中根据顾客需求出现的顺序在线安排任务进行加工,称这样的活动为实时调度.

供应链实时调度问题中,顾客随时都可能给制造商下达订单,制造商接到订单后,根据订单的紧急程度来安排生产加工,并将最终产品运送给顾客. 这些要加工的工件(称为临时工件)的到达时间、加工时间及数量等信息,在工件进入生产加工系统前是未知的,因为其信息的不确定性使得车间调度任务的难度大大增加. 在线调度问题中需要做出的决策为:如何安排临时工件的加工顺序及加工区间,使得这些工件在已有工件已经安排调度的情况下能尽早完成. Igor 等[1]从理论上研究了供应链的在线调度问题,其中顾客给制造商下达订单,加工完成后运送给顾客,其生产目标是最小化总费用. Chen 等[2,3]给出了一种带有时间窗的化工生产线调度问题的组合启发式算法. 文献[4]中 Fabrice 等提出了对实时加工系统中增加一个临时工件的在线调度问题. 李建祥等[5]研究了一类静态调度,从无缝钢管生产作业中提炼出的新的并行流水车间调度问题,且该调度具有工件无等待、工序之间存在运输时间等特点. 轩华等[6]提出了运用拉格朗日松弛算法求解实时无等待混合流水车间调度问题.

本文研究的这类实时系统是以金属表面的化学氧化工艺加工为背景. 金属部件采用化学浸渍方法处理,形成氧化膜层,每种不同的化学溶液使用不同的槽子来盛放,这些部件需要依次经过不同的化学溶液进行相关的表面处理. 这类表面处理系统的灵

收稿日期: 2007-07-13; **修回日期**: 2007-10-24.
基金项目: 国家自然科学基金项目(70671057, 70771052); 教育部博士点基金项目(20051065002); 山东省泰山学者计划; 青岛农业大学高层次人才启动基金.
作者简介: 常桂娟(1976—),女,山东荣成人,讲师,博士,从事供应链管理、生产调度等研究; 张纪会(1969—),男,山东临朐人,教授,博士生导师,从事复杂适应系统理论、物流与供应链管理等研究.

活性在于表面处理时间介于一个时间区间内,如果超出规定的时间区间则可能会导致金属表面的硬度、光泽、弹性等受到影响. 这类问题与传统的车间调度问题的区别在于:1) 工序的转移由机器人完成,转移时间不能忽略;2) 工序的加工时间介于一个区间;3) 工件可能随时进入生产系统,控制系统必须实时反应,对进入系统的每个工件要迅速有效地进行调度;4) 无等待约束要求工件的一道工序加工完后,必须立即转入下一道工序,除非是在工件转移过程中. 由此可知,一旦一个工件由控制系统进行调度,它就不允许重新调度. 因此,每个新到的工件的调度不能影响到其他正在调度的工件的加工.

本文研究了制造商接到一批临时产品订单需要及时加工时,如何合理安排这些临时工件的加工顺序以及合理利用资源的空闲时间,才能使得这些工件在不改变已有工件调度的情况下加工时间最短. 当临时加工工件数为 1 时,可以直接按文献[4]中提到的算法计算,但随着问题规模的增大,直接计算很困难. 基于此,本文给出了求解大规模订单的算法. 算法中使用微粒群算法对临时进入加工系统工件的加工顺序进行迭代,使问题在较短的时间内都能得到最优解.

2　微粒群算法

2.1　算法描述

微粒群优化算法(PSO)是一种基于群体智能理论的优化算法,由 Kennedy 等[7]提出. 该算法通过模拟鸟类群体调整自身飞行速度和飞行方向,将所有个体移动到适应度好的环境中,从而抽象出一种可以求解具有复杂解空间性质问题的优化算法. 传统微粒群算法的早期应用是在连续函数优化问题上展开的,其优化性能通过大量的实验已得到证实. 近几年,国内一些用微粒群算法解决车间调度问题的文献也开始陆续出现. 其中有夏蔚军等[8]提出的微粒群算法与模拟退火算法相结合的混合启发式算法;彭传勇等[9]提出的利用遗传算法交叉变异操作的思想,将粒子群算法与禁忌搜索相结合的广义粒子群优化算法.

在 n 维空间中有 N 个粒子,每个粒子的位置为 $X_i = (X_{i1}, X_{i2}, \cdots, X_{in})$,其中 $i = 1, 2, \cdots, N$,并具有与优化目标函数相关的适应度 Fit_i,同时每个粒子具有各自的速度 $V_i = (V_{i1}, V_{i2}, \cdots, V_{in})$. 对于粒子 i 所经历过的历史最好位置记为 $P_i = (P_{i1}, P_{i2}, \cdots, P_{in})$,称为 p_{best}. 群体所有粒子经历过的最好位置表示为 $P_g = (P_{g1}, P_{g2}, \cdots, P_{gn})$,称为 g_{best}. 微粒群算法描述如下:

$$V_{id}(t+1) = WV_{id}(t) + c_1 r_1 (P_{id}(t) - X_{id}(t)) + c_2 r_2 (P_{gd}(t) - X_{id}(t)), \quad (1)$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1). \quad (2)$$

其中:下标 d 表示粒子的维度;W 是惯性权值;c_1 和 c_2 均是正的常数,称为加速系数,合适的加速系数可以加快收敛而不易陷入局部最优,通常取为 2;r_1 和 r_2 是 $[0,1]$ 之间的随机数. 利用式(1) 和(2) 可计算第 $t+1$ 代粒子的速度和位置. 为了减少粒子离开搜索空间的可能性,通常将随机产生的初始粒子位置限定在 0 ～ 4 之间,速度限定在 - 4 ～ 4 之间[10].

微粒群算法流程如下:

Step1: 初始化一群粒子,随机产生每个粒子的位置和速度.

Step2: 评价每个粒子的适应度.

Step3: 对于每个粒子,将其适应度值与自身 p_{best} 比较,如果优于 p_{best},则将当前值设为该粒子的 p_{best}.

Step4: 对于全体粒子,将当前最优适应度值与 g_{best} 比较,如果优于 g_{best},则将当前最优适应度值设为群体 g_{best}.

Step5: 由式(1) 和(2) 计算每个粒子的新速度和位置.

Step6: 如未达到终止条件,则返回 Step2.

2.2　改进微粒群算法(IPSO)

2.2.1　编　　码

调度的关键问题是如何安排工件的加工顺序. 本文通过分析传统微粒群算法的优化机理,提出了一种基于粒子位置元素值排列编码的改进微粒群优化模型,并以此模型为基础构建了适合求解调度问题的改进微粒群优化算法. 假如有 L 个工件需要加工,将一个粒子表示为随机产生的 L 维向量,向量的 L 个元素依次对应着 L 个工件,粒子的维数就等于工件数. 按向量元素的大小对工件进行排列,向量元素值越小,其对应的工件就越早加工(对于具有相同值的元素,靠前元素所对应的工件优先加工). 例如,对于一个具有 6 工件的加工系统,随机产生一个 6 维粒子向量(2.40, 1.71, - 0.89, 3.10, - 2.34, - 1.20),粒子向量中元素最小值为 - 2.34,则其对应的工件 5 就最早加工. 依次类推,可以得到这样一组工件加工序列(5 6 3 2 1 4). 这种表示方法使得粒子向量中的元素与工件一一对应.

2.2.2　惯性权重

在微粒群优化算法中,惯性权重 W 是关系到 PSO 算法搜索能力的重要参数,将 W 从相对较大的值线性地减小到相对较小的值,使 PSO 算法在开始时具有很强的全局搜索能力,而在算法接近结束时

具有更好的局部搜索能力. 计算公式为

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{N_{\max}} N.$$

其中: $W_{\max}$ 和 $W_{\min}$ 为惯性权重的初始值和最终值, 分别取 1.2 和 0.4[8]; $N_{\max}$ 为最大迭代次数; N 为当前迭代次数.

3 在线调度问题

实时调度有两种不同的类型: 一类是在不改变以前调度的情况下安排任务加工, 称这种调度为使用空闲区间在线作业; 另一类是允许已存在的调度做一些有限的改变, 这种调度称为部分再调度的在线作业[11]. 本文考虑第一类.

本文讨论的问题是无等待(no-wait)约束, 即一个工件的前道工序结束后立即转入下一道工序的加工, 这里考虑了工序的运输时间. 在实时调度系统中, 资源可以使用的加工时间是一些时间区间值. 是否能在这些区间值上加工取决于要加工工件的加工时间及资源的空闲时间, 如果超过加工区间, 则会导致该资源上其他工件无法正常加工. 因此, 合理安排临时工件的加工顺序和加工时间至关重要.

3.1 问题描述

假设利用 m 种不同的化学溶液来处理金属零部件表面, 每种化学溶液分别有 n_i 个槽子, $i = 1,2,\cdots,m$. 用 a_{ij}^k 和 b_{ij}^k 分别表示盛放化学溶液 i 的槽子 j 的第 k 个允许加工区间的开始时间和结束时间. 用 $x_i\ (i = 1,2,\cdots,m)$ 表示工件在化学溶液 i 里开始浸泡的时间, x_{m+1} 表示最后一道工序的结束时间. 分别用 θ_i 和 $\theta_i + \delta_i$ 表示工件在槽子 i 内可以加工的时间下界和上界.

将盛放同一种化学溶液的槽子归为一类, 把它们的可以加工区间按 a_{ij}^k 递增的顺序排列, 如果区间左端点 a_{ij}^k 相同, 则按 b_{ij}^k 的递增顺序排列. 排列完毕, 得到一个加工时间序列 $S_i = \{(\alpha_{r_i}, \beta_{r_i})\}$, 其中 $r_i = 1,2,\cdots,q_i$, q_i 为盛放溶液 i 的槽子总数.

本文使用机器人来完成工件的转移, 机器人接收到调度指令后, 要立刻到达指定地点. 设到达第 i 种化学溶液的槽子所需要的时间为 π_i, $i = 1,2,\cdots,m$, 本文均取 1. 当工件的一道工序结束后, 机器人需携带工件立即转移, 设从第 i 个槽子转移到第 $i+1$ 个槽子所需的转移时间为 $w_{i,i+1}$, 本文均取 2. 机器人的可用区间用 $\{(\alpha_{0r_0}, \beta_{0r_0})\}$ 表示, 其中 $r_0 = 1,2,\cdots,q_0$, q_0 为机器人可用区间的总数.

这样, 问题就可以描述为对每个工件, 寻找开始加工时间序列 $X = \{x_1, x_2, \cdots, x_m, x_{m+1}\}$ 及资源(槽子和机器人统称为资源)的使用区间. 当加工完一个工件时, 应及时更新资源的可用时间区间, 目标是最小化所有工件的最大结束时间, 且满足如下关系式:

$$\theta_i \le x_{i+1} - x_i - w_{i,i+1} \le \theta_i + \delta_i,\quad i = 1,2,\cdots,m-1; \tag{3}$$

$$\theta_m \le x_{m+1} - x_m \le \theta_m + \delta_m; \tag{4}$$

$$x_i \ge \alpha_{ir_i},\ i = 1,2,\cdots,m; \tag{5}$$

$$x_{i+1} - \pi_i - w_{i,i+1} \ge \alpha_{0,z(i,i+1)},\quad i = 1,2,\cdots,m-1; \tag{6}$$

$$x_{i+1} - w_{i,i+1} \le \beta_{ir_i},\ i = 1,2,\cdots,m-1; \tag{7}$$

$$x_{m+1} \le \beta_{mr_m}; \tag{8}$$

$$x_{i+1} \le \beta_{0,z(i,i+1)},\ i = 1,2,\cdots,m-1. \tag{9}$$

其中 $z(i,i+1)$ 表示机器人从装有第 i 种溶液的槽子到第 $i+1$ 种溶液的槽子转移工件时使用的时间窗口. 上述关系式满足了无等待约束、临时工件只能在资源空闲时间内加工及加工时间介于 $[\theta_i, \theta_i + \delta_i]$ 内的要求.

3.2 寻找开始加工时间序列的算法描述

寻找工件开始加工时间序列 $X = \{x_1, x_2, \cdots, x_m, x_{m+1}\}$ 的算法如下:

$$t_1 = \alpha_{1r_1}; \tag{10}$$

$$u_{i-1,i} = \max(\alpha_{0,z(i-1,i)} + \pi_{i-1}, t_{i-1} + \theta_{i-1}),\quad i = 2,3,\cdots,m; \tag{11}$$

$$t_i = \max(\alpha_{i,r_i}, u_{i-1,i} + w_{i-1,i}),\quad i = 2,3,\cdots,m; \tag{12}$$

$$t_{m+1} = t_m + \theta_m; \tag{13}$$

$$x_{m+1} = t_{m+1}; \tag{14}$$

$$x_m = \max(t_m, x_{m+1} - \theta_m - \delta_m); \tag{15}$$

$$y_{i-1,i} = \max(u_{i-1,i}, x_i - w_{i-1,i}),\quad i = m, m-1,\cdots,2; \tag{16}$$

$$x_{i-1} = \max(t_{i-1}, y_{i-1,i} - \theta_{i-1} - \delta_{i-1}),\quad i = m, m-1,\cdots,2. \tag{17}$$

3.3 算法可行性

结论 1 通过该算法, 可以得到开始加工时间序列 $\{x_1, x_2, \cdots, x_m, x_{m+1}\}$. 该序列满足式(3) ~ (6), 且该序列中的 x_i 为满足式(3) ~ (6)的最早可能值.

证明 1) 由式(17)有

$$x_{i-1} + \theta_{i-1} = \max(t_{i-1} + \theta_{i-1}, y_{i-1,i} - \delta_{i-1}), \tag{18}$$

由式(11)和(12)可得

$$t_{i-1} + \theta_{i-1} \le t_i - w_{i-1,i}. \tag{19}$$

由式(18)和(19)可知

$$x_{i-1} + \theta_{i-1} \le \max(t_i - w_{i-1,i}, y_{i-1,i} - \delta_{i-1}).$$

由式(15)和(17)有 $x_i \ge t_i$, 所以

$$x_{i-1} + \theta_{i-1} \le \max(x_i - w_{i-1,i}, y_{i-1,i} - \delta_{i-1}). \tag{20}$$

再由式(16) 可知

$$y_{i-1,i} - \delta_{i-1} = \max(u_{i-1,i} - \delta_{i-1}, x_i - w_{i-1,i} - \delta_{i-1}) \leq \max(u_{i-1,i} - \delta_{i-1}, x_i - w_{i-1,i}).$$

由式(12) 可得 $t_i \geq u_{i-1,i} + w_{i-1,i}$,所以 $x_i \geq u_{i-1,i} + w_{i-1,i}$,进而 $x_i + \delta_{i-1} \geq u_{i-1,i} + w_{i-1,i}$,即 $x_i - w_{i-1,i} \geq u_{i-1,i} - \delta_{i-1}$,所以

$$y_{i-1,i} - \delta_{i-1} \leq x_i - w_{i-1,i}. \quad (21)$$

由式(20) 和(21) 可知

$$x_{i-1} + \theta_{i-1} \leq x_i - w_{i-1,i}, \quad (22)$$

由式(16) 可知

$$y_{i-1,i} \geq x_i - w_{i-1,i}, \quad (23)$$

再由式(17) 可得 $x_{i-1} \geq y_{i-1,i} - \theta_{i-1} - \delta_{i-1}$,结合式(23) 有

$$x_{i-1} \geq x_i - w_{i-1,i} - \theta_{i-1} - \delta_{i-1}. \quad (24)$$

综合式(22) 和(24) 可知式(3) 成立.

2) 由式(13) 和(14) 可知 $x_{m+1} = t_m + \theta_m$,由式(15) 有

$$x_m + \theta_m = \max(t_m + \theta_m, x_{m+1} - \delta_m) \leq (t_m + \theta_m, x_{m+1}) = x_{m+1}, \quad (25)$$

又有

$$x_m \geq x_{m+1} - \theta_m - \delta_m, \quad (26)$$

综合式(25) 和(26) 可知式(4) 成立.

3) 由式(12) 可知 $t_i \geq \alpha_{i,r_i}$,又因为 $x_i \geq t_i$,所以 $x_i \geq \alpha_{i,r_i}$, $i = 1,2,\cdots,m$,式(5) 成立.

4) 由式(11) 可知 $u_{i-1,i} \geq \alpha_{0,z(i-1,i)} + \pi_{i-1}$,由式(12) 有 $t_i \geq u_{i-1,i} + w_{i-1,i}$. 因为 $x_i \geq t_i$,所以 $x_i \geq \alpha_{0,z(i-1,i)} + \pi_{i-1} + w_{i-1,i}$, $i = 2,3,\cdots,m$, 式(6) 成立.

5) 序列中 x_i 为满足式(3) ～ (6) 的最早可能值. 因为 θ_i 为工件 i 最短加工时间,算法中 t_i 的取值不能减小,进而 x_i 也不能再减小,所以本算法所给出的开始加工时间均是最小可能值. □

结论 1 给出了由算法得到的序列 X 满足式(3) ～ (6) 的证明,此外,还需要满足式(7) ～ (9).

定义 1　如果对所有的 $i = 1,2,\cdots,m$,有 $v_i \leq r_i$ 且至少有一个 $i \in (1,2,\cdots,m)$,使得 $v_i < r_i$,则称 $S^* = \{(\alpha_{r_i}, \beta_{r_i})\}$ 为 $S = \{(\alpha_{v_i}, \beta_{v_i})\}$ 后面的序列.

结论 2　假设使用上述算法得到一个时间序列 $\{x_1, x_2, \cdots, x_m, x_{m+1}\}$, $S = \{(\alpha_{v_i}, \beta_{v_i})\}$ 为化学溶液允许使用的时间区间集,若存在 $k \in \{1,2,\cdots,m\}$,使得 $x_{k+1} - w_{k,k+1} > \beta_{k,r_k}$,则包含允许加工窗口 $(\alpha_{k,r_k}, \beta_{k,r_k})$ 且在 S 后面的序列 S^* 中不存在问题的可行解 X^*.

证明　因为 S^* 是 S 后面的序列,所以 $\alpha^*_{r_i} \geq \alpha_{i,r_i}$,对时间区间集 S^* 和 S,应用算法(10) ～ (13) 计算所得的 t_i^* 值不小于 t_i 值,进而有 $x_i^* \geq x_i$,所以

$$x^*_{k+1} - w_{k,k+1} \geq x_{k+1} - w_{k,k+1} > \beta_{k,r_k}.$$

由此可知, X^* 不是可行解. □

结论 2 表明,如果 S 中某一个时间区间 $(\alpha_{k,r_k}, \beta_{k,r_k})$ 不满足式(7) ,则用包含该区间的 S 的任何后面序列构造的开始加工时间序列均不是可行解. 当条件(7) 不满足时,必须重新寻找资源的可加工区间集. 利用该加工区间 $(\alpha_{k,r_k}, \beta_{k,r_k})$ 后面的区间替换该区间,其他区间不变化,然后使用上述算法重新构造时间序列 X. 同理,对于机器人也使用该方法.

下面给出求解在线调度问题的实时算法,随着临时工件规模的增大,为尽早完成这些临时订单任务,算法中使用了微粒群算法对临时进入加工系统的工件的加工顺序进行迭代.

3.4　实时调度算法

Step1: 通过对问题及解的特性进行分析和了解,设定参数、粒子群体规模及最大迭代次数.

Step2: 假设有 N 个临时工件,采用随机产生粒子位置和速度的方法初始化粒子群,由粒子位置可得到工件的加工顺序,按初始加工顺序,选出要进入加工系统的工件.

Step3: 对于每种化学溶液,给出如上所述的加工区间序列 $S_i = \{(\alpha_{r_i}, \beta_{r_i})\}$ $(r_i = 1,2,\cdots,q_i,\ i = 1,2,\cdots,m)$ 及机器人的可用时间区间集 $(\alpha_{r_0}, \beta_{r_0})$ $(r_0 = 1,2,\cdots,q_0)$.

Step4: 对于每个 $i = 1,2,\cdots,m$,令 $r_i = 1$;对于每个 $i = 2,3,\cdots,m$, 令 $z(i-1,i) = 1$.

Step5: 使用下面等式构造一个时间序列:

$$X = \{x_1, x_2, \cdots, x_{m+1}\},\ t_1 = \alpha_{1,r_1};$$

$$u_{i-1,i} = \max(\alpha_{0,z(i-1,i)} + \pi_{i-1}, t_{i-1} + \theta_{i-1}),\quad i = 2,3,\cdots,m;$$

$$t_i = \max(\alpha_{i,r_i}, u_{i-1,i} + w_{i-1,i}),\quad i = 2,3,\cdots,m;$$

$$t_{m+1} = t_m + \theta_m,\ x_{m+1} = t_{m+1};$$

$$x_m = \max(t_m, x_{m+1} - \theta_m - \delta_m);$$

$$y_{i-1,i} = \max(u_{i-1,i}, x_i - w_{i-1,i}),\quad i = m, m-1,\cdots,2;$$

$$x_{i-1} = \max(t_{i-1}, y_{i-1,i} - \theta_{i-1} - \delta_{i-1}),\quad i = m, m-1,\cdots,2.$$

Step6: 如果 $x_{m+1} \leq \beta_{m,r_m}$, $x_i \leq \beta_{0,z(i-1,i)}$ 与 $y_{i-1,i} \leq \beta_{i-1,r_{i-1}}$ 对于所有的 $i = 2,3,\cdots,m$ 成立,则转入 Step8,否则转至 Step7.

Step7: 如果 $x_{m+1} \geq \beta_{m,r_m}$,则令 $r_m = r_m + 1$;如果 $x_i > \beta_{0,z(i-1,i)}$ $(i = 2,3,\cdots,m)$,则令 $z(i-1,i) = z(i-1,i) + 1$;如果 $y_{i-1,i} > \beta_{i-1,r_{i-1}}$ $(i = 2,3,\cdots,m)$,则令 $r_{i-1} = r_{i-1} + 1$. 返回 Step5.

Step8:更新槽子及机器人的可用区间序列,并按本文方法重新排列可用时间区间,得到槽子及机器人的新的时间区间序列.

Step9: 按工件的加工顺序,选出下一个进入加工系统的工件,转入 Step4;若所有工件均加工完成,则转入 Step10.

Step10: 根据每个工件的完工时间,计算目标函数值,并记录全局最优粒子 P_g 及个体最优粒子 P_i.

Step11: 对于 N 个粒子,利用微粒群优化公式

$$V_{id}(t+1) = WV_{id}(t) + c_1 r_1 (P_{id}(t) - X_{id}(t)) + c_2 r_2 (P_{gd}(t) - X_{id}(t)),$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1),$$

产生新一代粒子群的位置及速度,并对全局最优粒子和个体最优粒子进行更新.

Step12: 如果满足最大迭代次数,则输出最优适应度函数值并终止;否则,返回 Step3.

由结论 1 和结论 2 可知,使用该算法所得到的每个工件的开始加工时间序列均为满足式(3) ~ (9) 的最优解.

4 算 例

算例 1 临时工件需要在 3 种化学溶液中浸泡,这 3 种化学溶液分别盛放在不同槽子中. 盛放第 1 种溶液的槽子有 3 个,盛放第 2 种溶液的槽子有 4 个,盛放第 3 种溶液的槽子有 3 个. 这些槽子可以使用的加工时间区间分别为:1) 第 1 种化学溶液:槽子 1 为[0,7],[10,17],[20,+∞);槽子 2 为[3,5],[8,12],[17,27],[30,+∞);槽子 3 为[2,6],[7,13],[15,18],[25,+∞). 2) 第 2 种化学溶液:槽子 1 为[2,7],[10,16],[21,+∞);槽子 2 为[0,6],[9,16],[20,27],[30,+∞);槽子 3 为[1,5],[8,13],[25,+∞);槽子 4 为[5,9],[15,+∞). 3) 第 3 种化学溶液:槽子 1 为[5,12],[13,19],[20,+∞);槽子 2 为[3,9],[12,27],[35,+∞);槽子 3 为[3,8],[9,15],[17,29],[30,+∞).

对于每种化学溶液,将其可加工区间按递增的顺序排列,得:

1) 化学溶液 I:

S_1 =
{[0,7],[2,6],[3,5],[7,13],[8,12],
[10,17],[15,18],[17,27],[20,+∞),
[25,+∞),[30,+∞)}.

2) 化学溶液 II:

S_2 =
{[0,6],[1,5],[2,7],[5,9],[8,13],
[9,16],[10,16],[15,+∞),[20,27],
[21,+∞),[25,+∞),[30,+∞)}

3) 化学溶液 III:

S_3 =
{[3,8],[3,9],[5,12],[9,15],[12,27],
[13,19],[17,29],[20,+∞),[30,+∞),
[35,+∞)}.

第 1 个机器人的空闲时间为[3,7],[12,15],[20,+∞). 第 2 个机器人的空闲时间为[2,6],[7,9],[13,18],[24,+∞). 将机器人可利用的空闲时间重新排列得[2,6],[3,7],[7,9],[12,15],[13,18],[20,+∞),[24,+∞). 3 个临时工件的 3 道工序加工时间区间为工件 1 是[3,5],[3,4],[4,6];工件 2 是[2,4],[3,4],[5,6];工件 3 是[6,7],[5,6],[2,4].

使用本文提出算法对该算例用计算机仿真,初始种群大小根据问题的规模来设定,当临时工件规模较小时,可选取较少的粒子进行迭代. 针对算例 1,种群规模设为 10,迭代次数设为 10. 仿真结果显示最短加工时间为 36. 通过仿真试验可以发现,10 个初始粒子在迭代 10 次之后,均得到生产周期为 36 的仿真结果,由此可见本文算法对在线调度问题表现出了良好的计算能力.

下面给出其中一个最优调度结果:工件加工顺序为{3,2,1}. 工件 3 在 3 个槽子中的加工时间段分别为[7,13],[15,21],[23,25],使用机器人的时间段分别为[12,15],[20,23];工件 2 在 3 个槽子中的加工时间段分别为[20,24],[26,29],[31,36],使用机器人的时间段分别为[23,26],[28,31];工件 1 在 3 个槽子中的加工时间段分别为[20,25],[27,30],[32,36],使用机器人的时间段分别为[24,27],[29,32].

算例 2 为进一步验证实时调度算法的有效性,本例给出了较大规模的在线调度问题. 临时加工工件数为 10,每个工件有 6 道工序,即有 6 种化学溶液. 每种化学溶液可能有多个槽子来盛放,下面直接给出盛放不同溶液的槽子和机器人的可用时间区间.

化学溶液可加工时间区间分别为

S_1 =
{[0,7],[2,5],[6,10],[7,13],
[8,15],[10,16],[13,19],[17,25],
[21,29],[30,37],[39,+∞),
[40,+∞),[45,+∞),[49,+∞)};

S_2 =
{[2,6],[3,7],[4,9],[7,10],[9,14],

$[12,19],[15,25],[20,26],[27,30],$
$[32,47],[42,49],[43,+\infty),$
$[50,+\infty),[53,+\infty),[55,+\infty)\};$
$S_3 =$
$\{[4,10],[5,9],[5,13],[7,19],[11,16],$
$[12,17],[18,29],[30,+\infty),[35,+\infty),$
$[39,+\infty),[42,+\infty),$
$[44,+\infty),[47,+\infty)\};$
$S_4 =$
$\{[3,7],[4,7],[4,9],[7,16],[8,12],$
$[10,18],[14,19],[20,26],[29,36],$
$[32,39],[42,59],[55,+\infty),$
$[57,+\infty),[60,+\infty),[62,+\infty)\};$
$S_5 =$
$\{[4,8],[5,14],[7,14],[9,12],$
$[10,16],[13,19],[18,26],[21,32],$
$[31,39],[40,47],[49,+\infty),$
$[53,+\infty),[58,+\infty),[60,+\infty)\};$
$S_6 =$
$\{[7,19],[8,13],[8,18],[9,15],$
$[13,19],[21,28],[24,30],[30,37],$
$[32,40],[43,57],[53,+\infty),[60,+\infty),$
$[63,+\infty),[69,+\infty),[80,+\infty)\}.$

机器人可以使用的时间区间为

$\{[6,15],[13,24],[15,30],[26,37],$
$[30,50],[40,55],[51,67],[52,66],$
$[65,77],[72,88],[80,90],[85,99],[100,120],$
$[101,+\infty),[108,+\infty),$
$[120,+\infty),[125,+\infty)\}.$

临时工件的加工时间区间分别为

工件 1:$[2,4],[3,4],[3,4],[2,3],[4,5],[3,5];$
工件 2:$[2,3],[3,5],[2,3],[2,3],[4,5],[4,6];$
工件 3:$[2,4],[4,5],[2,3],[2,3],[3,4],[2,4];$
工件 4:$[3,5],[2,4],[3,4],[2,3],[3,4],[3,5];$
工件 5:$[4,5];[2,4];[2,3];[2,4];[3,5];[3,4];$
工件 6:$[2,4],[3,5],[4,5],[2,4],[4,6],[3,4];$
工件 7:$[4,5],[3,5],[2,4],[2,3],[2,3],[2,4];$
工件 8:$[3,4],[3,4],[3,4],[2,4],[3,5],[3,4];$
工件 9:$[3,5],[3,4],[2,3],[3,4],[2,4],[3,4];$
工件 10:$[4,5],[3,4],[2,3],[2,4],[3,4],[3,5].$

对于算例 2,种群规模取为 20,迭代次数为 60,多次仿真结果显示,最短加工时间为 123. 下列矩阵 A 记录了取得最优解时,加工顺序为$[10\ 9\ 3\ 1\ 8\ 6\ 2\ 5\ 4\ 7]$ 的工件在每个槽子中加工的起始时间(最后一列分别记录了每个工件的结束时间).

$$A = \begin{bmatrix} 4 & 9 & 15 & 20 & 24 & 30 & 34 \\ 21 & 27 & 32 & 37 & 41 & 46 & 49 \\ 11 & 17 & 24 & 29 & 33 & 38 & 40 \\ 39 & 43 & 48 & 54 & 58 & 64 & 67 \\ 42 & 49 & 54 & 58 & 62 & 67 & 70 \\ 48 & 55 & 61 & 65 & 70 & 74 & 77 \\ 69 & 75 & 80 & 85 & 89 & 95 & 98 \\ 70 & 77 & 83 & 88 & 92 & 98 & 101 \\ 96 & 103 & 107 & 111 & 115 & 120 & 123 \\ 97 & 104 & 109 & 113 & 117 & 121 & 123 \end{bmatrix}.$$

5　结　　论

本文给出了在不改变已有工件调度的情况下,尽早完成临时订单的算法. 提出了基于粒子位置元素值排列的编码方法,将应用于连续问题的微粒群算法成功应用于调度问题,使得问题在较大规模的情况下,使用较少的种群及进化代数同样能迅速找到最优加工方案,完成临时订单的加工,对制造商的实际生产供应链管理具有一定的指导意义. 第 2 类实时调度即部分再调度的在线作业问题,将是下一步研究的方向.

参考文献(References)

[1] Igor Averbakh, Xue Z H. On-line supply chain scheduling problems with preemption[J]. European J of Operational Research, 2007, 181(1): 500-504.

[2] Chen H X, Chu C, Proth J M. Sequencing of parts in robotic cells [J]. Int J of Flexible Manufacturing Systems, 1997, 9(1): 81-103.

[3] Chen H X, Chu C, Proth J M. Cyclic scheduling of a hoist with time windows constraints[J]. IEEE Trans on Robotics and Automation, 1998, 14(1): 144-152.

[4] Fabrice Chauvet, Eugene Levner, Lenoid K Meyzin, et al. On-line scheduling in a surface treatment system [J]. European J of Operational Research, 2000, 120(2): 382-392.

[5] 李建祥, 唐立新, 吴会江. 带运输和设置时间的无等待并行流水车间调度问题研究[J]. 系统工程理论与实践, 2006, 26(1): 18-24.
(Li J X, Tang L X, Wu H J. No-wait parallel flowshop scheduling with transfer and setup times [J]. Systems Engineering — Theory and Practice, 2006, 26(1): 18-24.)

[6] 轩华, 唐立新. 实时无等待 HFS 调度的一种拉格朗日松弛算法[J]. 控制与决策, 2006, 21(4): 376-380.
(Xuan H, Tang L X. Lagrangian relaxation algorithm for real-time hybrid flow-shop scheduling with no-wait in process[J]. Control and Decision, 2006, 21(4): 376-380).

(下转第 1102 页)

M-SIMPSA 和 MIHDE 中的个别算例外,HEA 对适应度函数的平均计算次数明显少于其他算法.

5 结 论

本文把统计试验设计的方法——正交设计作为交叉算子,给出了一种混合启发式的变异算子.为了提高种群的多样性,引入一种迁移算子,提出一种混合进化算法.本文算法不仅能求解混合整数规划,而且能求解纯整数规划.数值实验结果表明,所提出的混合进化算法是快速而有效的,同已有的其他算法的数值结果相比,本文算法计算量小,求得解的精度高.

参考文献(References)

[1] 孟志青,胡奇英,杨晓琪. 一种求解整数规划与混合整数规划非线性罚函数方法[J]. 控制与决策,2002,17(3):310-314.
(Meng Z Q, Hu Q Y, Yang X Q. A method of nonlinear penalty function for solving integer programming and mixed integer programming [J]. Control and Decision, 2002, 17(3): 310-314.)

[2] 肖建,张志宏. 一种求非线性整数规划全局最小解的算法[J]. 石家庄学院学报,2006,8(6):49-53.
(Xiao J, Zhang Z H. An algorithm for solving the global optimization of nonlinear integer programming [J]. J of Shijiazhuang University, 2006, 8(6): 49-53.)

[3] Lino Costa, Pedro Oliveira. Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems [J]. Computers and Chemical Engineering, 2001, 25(2): 257-266.

[4] Lin Y C, Hwang K S. A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems[J]. Computers and Mathematics with Applications, 2004, 47(8): 1295-1307.

[5] 方开泰,马长兴. 正交与均匀试验设计[M]. 北京:科学出版社,2001.
(Fang K T, Ma C X. Orthogonal and uniform experimental design [M]. Beijing: Science Press, 2001.)

[6] Leung Y W, Wang Y. An orthogonal genetic algorithm with quantization for global numerical optimization[J]. IEEE Trans on Evolutionary Computation, 2001, 5(1): 41-53.

[7] Ho S Y, Shu L S, Chen J H. Intelligent evolutionary algorithms for large parameter optimization problems [J]. IEEE Trans on Evolutionary Computation, 2004, 8(6): 522-540.

[8] Tsai J T, Liu T K, Chou J H. Hybrid taguchi-genetic algorithm for global numerical optimization [J]. IEEE Trans on Evolutionary Computation, 2004, 8(4): 365-377.

[9] Zhang Q, Leung Y W. An orthogonal genetic algorithm for multimedia multicast routing [J]. IEEE Trans on Evolutionary Computation, 1999, 3(1): 53-62.

[10] Li H, Jiao Y C, Zhang L, et al. Genetic algorithm based on the orthogonal design for multidimensional knapsack problems [C]. Proc of 2nd Int Conf on Advances in Natural Computation. Birlin: Springer-Verlag, 2006, 1: 696-705.

(上接第 1097 页)

[7] Kennedy J, Eberhart R. Particle swarm optimization [C]. IEEE Int Conf on Neural Networks. Perth, 1995: 1942-1948.

[8] 夏蔚军,吴智铭. 基于混合微粒群优化的多目标柔性 Job-shop 调度[J]. 控制与决策,2005,20(2):137-141.
(Xia W J, Wu Z M. Hybrid particle swarm optimization approach for multi-objective flexible job-shop scheduling problems[J], Control and Decision, 2005, 20 (2): 137-141.)

[9] 彭传勇,高亮,邵新宇,等. 求解作业车间调度问题的广义粒子群优化算法[J]. 计算机集成制造系统,2006,12(6):911-917.
(Peng C Y, GAO L, Shao X Y, et al. General particle swarm optimization algorithm for job-shop scheduling problem [J]. Computer Integrated Manufacturing Systems, 2006, 12(6): 911-917).

[10] Fatih Tasgetiren M, Yun-Chia Liang, Mehmet Sevkli, et al. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem [J]. European J of Operational Research, 2007, 177(3): 1930-1947.

[11] Jean-Marie Proth. Scheduling: New trends in industrial environment[J]. Annual Reviews in Control, 2007, 31(1): 157-166.