文章编号: 1001-0920 (2012) 07-1077-05

粒子群算法在Lot-sizing问题中的应用

闫 萍[1], 焦明海[2], 赵冰梅[1]

(1. 沈阳航空航天大学 经济与管理学院，沈阳 110136；2. 东北大学 计算中心，沈阳 110819)

摘 要: 针对无能力限制的Lot-sizing问题, 提出一种改进的离散粒子群优化算法. 设计粒子编码为生产设备的调整状态, 通过有效的解码程序将粒子解释为生产计划. 区别于传统的粒子群算法, 算法采用单切点交叉算子来提高算法的局部求精能力, 并引入变异算子和速度扰动策略保持种群的多样性, 使算法在局部求精和空间探索间取得了较好的平衡. 在随机生成的90组测试实例中对算法性能进行仿真实验, 结果表明该算法具有良好的性能.

关键词: 生产计划；Lot-sizing问题；粒子群优化；遗传算子

中图分类号: TP18 **文献标识码:** A

Application of particle swarm optimization algorithm in Lot-sizing problem

YAN Ping[1], *JIAO Ming-hai*[2], *ZHAO Bing-mei*[1]

(1. School of Economics and Management，Shenyang Aerospace University，Shenyang 110136，China；2. Computer Center，Northeastern University，Shenyang 110819，China. Correspondent：YAN Ping，E-mail：yanping@sau.edu.cn)

Abstract: An improved discrete particle swarm optimization(PSO) algorithm is designed to tackle the general uncapacitated Lot-sizing problem. The encoding scheme of particles is designed in terms of setup states of production units, while an effective decoding procedure translates a particle into a feasible production plan. Different from the traditional PSO algorithm, the improved PSO algorithm incorporates single cutting-point crossover operators to improve the intensification ability of the algorithm. In addition, mutation operators and velocity disturbance strategies are also introduced into the PSO algorithm to keep the diversity of swarm. By using those operators, the proposed algorithm can get good balance between exploitation and exploration. Computational results on 90 randomly generated test instances show the good performance of the proposed PSO algorithm.

Key words: production planning；Lot-sizing problem；particle swarm optimization；genetic operator

1 引 言

Lot-sizing问题是一类经典的生产计划问题[1-2]. 该问题可描述为在给定的产品需求模式下, 制定出最佳的生产方案, 使得设备调整和产品库存的总费用最小. 在生产过程中, 减少设备的调整次数将会降低调整费用, 但产品的库存费用会增加; 相反, 如果采用按需生产的策略, 增加生产的调整次数, 虽然降低了库存的费用, 但同时也增加了调整费用. 因此, 在调整费用和库存费用之间存在着一种平衡, 成为求解这类问题的难点. Lot-sizing问题广泛存在于工业生产的各个领域, 对于该问题的研究有利于企业降低产品生产成本, 提高生产效率, 所以对它的研究越来越受到人们的重视.

近年来, 学者们相继提出一些启发式方法和智能优化算法, 取得了一些成果. Xie等[3]提出一种基于启发式的遗传算法求解这类问题. Xiao等[4]采用变邻域搜索算法解决一类多级多产品的Lotsizing问题, 但当问题规模较大时, 仍存在求解速度较慢或计算结果较差的缺点. 粒子群优化(PSO)算法作为一种基于群智能的优化技术在许多Lot-sizing问题中都取得了较好的应用. Han等[5]应用PSO算法求解一类带有集成产品结构特征的多级Lot-sizing问题. 针对集成产品结构特征, 算法对速度和位置更新公式以及相关的运算符进行了重新定义. Dye等[6]提出一类

收稿日期: 2010-12-19；**修回日期:** 2011-05-17.

基金项目: 国家自然科学基金青年基金项目(71001074)；沈阳航空航天大学博士启动基金项目(11YB11).

作者简介: 闫萍(1980−), 女, 讲师, 博士, 从事生产计划与调度问题的研究；焦明海(1973−), 男, 副教授, 博士, 从事供应链与物流优化理论的研究.

确定性经济订货批量模型, 通过设计有效的粒子编码方案, 应用PSO算法成功求解了该模型. 本文考虑将遗传算子引入PSO求解无能力限制的Lot-sizing问题. 目前, 已有一些学者对带有遗传算子的PSO算法进行了相关研究. Arumugam等[7]将算术交叉、平均凸交叉和根概率交叉3种算子引入PSO中以提高算法的收敛速度, 并通过求解一类约束优化控制问题验证了算法的性能. Coelho[8]设计一种带有混沌变异算子的量子粒子群算法, 有效求解了一类机械工程设计问题. Tavakkoli等[9]采用多目标PSO算法求解Jobshop问题, 将遗传算子作为变邻域搜索方法, 较好地避免了算法的早熟收敛. 上述算法将不同的交叉、变异算子嵌入PSO, 都取得了较好的性能.

为了提高标准PSO算法的收敛速度和寻优精度, 本文提出一种克服早熟的改进PSO算法, 将单切点交叉、变异算子引入PSO中. 通过分析Lot-sizing问题的特征, 挖掘出一些最优解的性质用于指导PSO的求解过程. 提出的改进PSO算法采用交叉搜索算子改善算法的局部求精能力, 并引入变异算子和速度扰动策略增加粒子种群的多样性, 较好地避免了种群早期收敛、后期停滞不前的现象.

2 Lot-sizing问题描述

无能力限制的Lot-sizing问题可以描述如下: 设备在由T个时间周期组成的计划区间内生产产品, 产品的需求d_t随时间周期t发生变化; 设备的生产能力没有限制并且不允许缺货; 设备在每次运行前, 需要对其进行调整, 调整费用为sc; 每个时间周期内的剩余产品存入仓库, 单位产品的库存保管费用为h, 并且仓库的存储能力没有限制; 在满足产品计划需求约束的条件下, 制定生产计划, 安排设备的生产运行时间和每次生产的产量, 使得设备的调整费用与仓库的库存费用之和最小. 设问题的决策变量: X_t为设备在时间周期t内是否发生调整, Y_t为设备在时间周期t内的产量, I_t为产品在时间周期t末的库存量. Lot-sizing问题的数学模型如下:

$$\min \quad \sum_{t=1}^{T} sc\cdot X_t+\sum_{t=1}^{T} h\cdot I_t. \tag{1}$$

$$\text{s.t.} \quad I_{t-1}+Y_t=d_t+I_t,\ t=1,2,\cdots,T; \tag{2}$$

$$Y_t\leqslant\sum_{t=1}^{T} d_t\cdot X_t,\ t=1,2,\cdots,T; \tag{3}$$

$$X_t=\begin{cases}1\ ,\ Y_t>0,\\ 0\ ,\ Y_t=0,\end{cases}\quad Y_t\geqslant 0;$$

$$I_t\geqslant 0,\ I_0=0,\ t=1,2,\cdots,T. \tag{4}$$

目标函数(1)为最小化设备的调整费用与产品的库存费用之和, 约束条件(2)为产品的需求量、产量与库存量之间的物料平衡约束, 约束条件(3)和(4)描述了生产变量Y_t与调整变量X_t之间的关系以及各决策变量的取值范围.

3 标准粒子群优化算法

PSO算法是模拟动物群体捕食行为的仿生算法. 该算法初始化为一群随机粒子. 每个粒子都有一个速度决定其飞行方向和距离. 在每次迭代中, 粒子通过跟踪两个"极值"来更新自己: 一个是粒子本身找到的最好解, 称为个体极值点; 另一个是整个种群目前找到的最好解, 称为全局极值点. 粒子的优劣由被优化的适应值函数来衡量.

设在一个N维的搜索空间, PSO算法中第i个粒子的速度和位置分别用向量$V_i=(v_{i1},v_{i2},\cdots,v_{iN})$和$X_i=(x_{i1},x_{i2},\cdots,x_{iN})$表示, 其中$i=1,2,\cdots$, Popsize, Popsize为种群的粒子数. 粒子i迄今为止搜索到的最好位置为$P_i=(p_{i1},p_{i2},\cdots,p_{iN})$, 整个种群目前找到的最好位置为$P_{\text{gbest}}=(p_{\text{gbest}1},p_{\text{gbest}2},\cdots,p_{\text{gbest}N})$. 粒子$i$的速度和位置的更新方程如下[10]:

$$v_{ij}^{k+1}=wv_{ij}^{k}+c_1r_1(p_{ij}^{k}-x_{ij}^{k})+ c_2r_2(p_{\text{gbest}j}^{k}-x_{ij}^{k}), \tag{5}$$

$$x_{ij}^{k+1}=x_{ij}^{k}+v_{ij}^{k+1}. \tag{6}$$

其中: k为算法的迭代次数, w为惯性系数, c_1和c_2为加速系数, 分别调节向个体最好粒子和全局最好粒子方向飞行的最大步长. 恰当的c_1和c_2取值可以实现在加快收敛速度的同时不易陷入局部最优点, 通常令$c_1=c_2=2$; r_1和r_2是[0, 1]之间的随机实数.

4 求解Lot-sizing问题的离散粒子群优化算法

4.1 粒子的编码与解码方法

在无能力限制Lot-sizing问题的最优解中存在如下性质[11]:

命题 1 在无能力限制的Lot-sizing问题中, 存在一个最优的生产计划满足方程$Y_t\cdot I_{t-1}=0$.

利用这个性质, 本文将粒子的位置X定义为由0-1值组成的T维向量, 用下式表示:

$$X_i=(x_{i1},x_{i2},\cdots,x_{ij},\cdots,x_{iT}),x_{ij}\in\{0,1\},$$
$$i=1,2,\cdots,\text{Popsize}. \tag{7}$$

其中维表示时间周期, 每维数据表示设备的生产调整状态: 取值为1时表示设备发生调整; 取值为0时表示设备未发生调整.

粒子的解码方法为: 通过粒子的位置向量编码 X, 可以确定设备的生产时间周期和调整次数. 考虑到不允许缺货的情况发生, 利用性质 1, 可以推导出生产变量

$$Y_t = \begin{cases} \sum_{j=t}^{a(t)-1} d_j, \ X_t = 1; \\ 0, \ X_t = 0; \end{cases}$$

$$a(t) = \min\{t' | X_{t'} = 1 \text{ and } t < t' \leqslant T\}. \quad (8)$$

进而, 通过约束 (2) 可以计算出库存变量 $I_t = \sum_{j=1}^{t}(Y_j - d_j)$. 这样, 粒子的位置向量 X 可间接地表示 Lot-sizing 问题的一个可行的生产计划.

此外, 由于产品的初始库存 $I_0 = 0$, 并且第 1 个时间周期的需求量 $d_1 > 0$, 所以设备在第 1 个时间周期必须发生调整, 即 $X_1 = 1, Y_1 > 0$. 由该结论可知, 粒子群算法在优化的过程中仅需考虑 2～T 之间的时间周期内设备的生产调整状态, 避免了不可行解的出现并简化了计算.

4.2　粒子的速度

速度的作用是改变粒子的位置. 与粒子位置 X 的定义类似, 本文将速度 V 定义为 T 维实值向量, 用下式表示:

$$V_i = (v_{i1}, v_{i2}, \cdots, v_{ij}, \cdots, v_{iT}),$$

$$v_{ij} \in [-V_{\max}, V_{\max}], i = 1, 2, \cdots, \text{Popsize}. \quad (9)$$

其中每维数据表示设备处于某种调整状态的概率: 较大的 v_{ij} 值说明设备存在较大的概率处于生产调整状态 (即 $x_{ij} = 1$); 反之, 较小的 v_{ij} 值则倾向于设备处于未发生调整的状态 (即 $x_{ij} = 0$).

下面的例子进一步说明了粒子位置与速度的更新过程. 假设粒子的速度更新公式 (5) 中仅存在认知部分并且 $c_1 = r_1 = 1$, 粒子 i 的当前位置 X_i 和个体最好位置 P_i 如图 1 所示. 速度向量的各分量借助 sigmoid 函数[12](sigmoid$(x) = 1/[1+\exp(-x)]$) 转化为引导位置变化的概率. 当粒子的速度转化为概率值后,

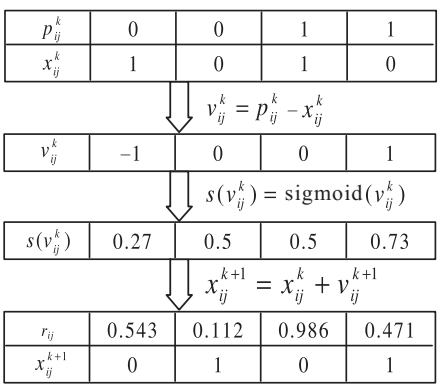

图 1　粒子位置的更新过程

对于每一维都产生一个 [0,1] 的随机实数 r_{ij} 用以更新粒子的位置: 如果 $r_{ij} \leqslant s(v_{ij}^k)$, 则 $x_{ij}^{k+1} = 1$; 否则, $x_{ij}^{k+1} = 0$.

4.3　粒子的评价

粒子的适应值函数作为评价粒子优劣的度量工具, 通常是所优化问题的目标函数. 在求解 Lot-sizing 问题的 PSO 算法中, 考虑到粒子的编码方法, 本文提出如下性质:

命题 2　在 Lot-sizing 问题的模型 (1)～(4) 中, 目标函数可以简化成

$$sc \cdot \sum_{t=1}^{T} X_t + h \cdot \sum_{t=1}^{T}(T - t + 1)(Y_t - d_t). \quad (10)$$

证明　Lot-sizing 问题的目标函数为设备的调整费用与产品的库存费用之和. 对于产品的总库存费用, 可以采用如下推导过程:

$$h \cdot \sum_{t=1}^{T} I_t = h \cdot \sum_{t=1}^{T}\sum_{j=1}^{t}(Y_j - d_j) =$$

$$h \cdot \sum_{t=1}^{T}[(Y_1 - d_1) + (Y_2 - d_2) + \cdots + (Y_t - d_t)] =$$

$$h \cdot \{(Y_1 - d_1) + [(Y_1 - d_1) + (Y_2 - d_2)] + \cdots +$$

$$[(Y_1 - d_1) + (Y_2 - d_2) + \cdots + (Y_T - d_T)]\} =$$

$$h \cdot [T \cdot (Y_1 - d_1) + (T-1) \cdot (Y_2 - d_2) + \cdots + (Y_T - d_T)] =$$

$$h \cdot \sum_{t=1}^{T}(T - t + 1)(Y_t - d_t).$$

由上可知, 目标函数 (1) 可以简化成式 (10). □

与目标函数式 (1) 相比, 式 (10) 中省略了对库存变量 I_t 的求解. 当粒子的位置向量确定后, 可根据式 (10) 评价粒子的优劣, 其中生产变量 Y_t 通过式 (8) 计算获得.

4.4　交叉算子

为了提高 PSO 的局部求精能力, 本文将单切点交叉算子嵌入 PSO 进化过程, 切点的位置随机产生. 交叉算子的父代分别是粒子的当前位置向量 X_i 和种群中随机选择的粒子位置向量 $X_g (g \in \{1, 2, \cdots, \text{Popsize}\}, g \geqslant i)$. 每个父代粒子的位置向量被切点分为左右两部分: X_i 在切点左侧部分的位置、速度与 X_g 在切点右侧部分的位置、速度组合成子代粒子 A_1; X_g 在切点左侧部分的位置、速度与 X_i 在切点

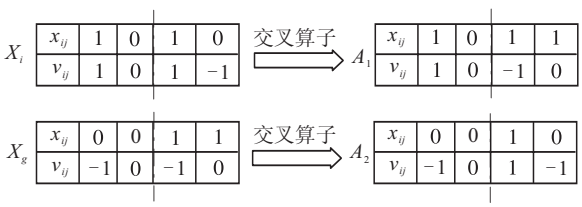

图 2　粒子群算法中的交叉算子

右侧部分的位置、速度组合成子代粒子 A_2. 如果生成的子代粒子 A_1 或 A_2 优于父代粒子 X_i, 则用最好的子代粒子更新X_i. 交叉算子的运算过程如图 2 所示.

4.5 变异算子和局部最优逃逸策略

当粒子 $X_i = (x_{i1}, x_{i2}, \cdots, x_{iT})$ 执行变异算子时, 首先随机选择一个变异点 λ ($\lambda \in \{1, 2, \cdots, T\}$). 如果 $x_{i\lambda} = 1$, 则变异后的 $x_{i\lambda}$ 取值为 0; 反之, 如果 $x_{i\lambda} = 0$, 则变异后的 $x_{i\lambda}$ 取值为 1. 为了避免粒子群陷入局部最优点, 本文设计了一个速度扰动策略. 如果算法获得的全局最好解的适应值连续 G 代都相同, 则对当前的粒子种群应用速度扰动策略; 从种群中随机选择一个粒子, 对其速度向量的取值重新初始化.

4.6 算法实现过程

求解 Lot-sizing 问题的改进 PSO 算法的具体流程如下:

Step 1: 设置粒子群优化算法的参数.

Step 2: 初始化粒子群. 在 $\{0, 1\}$ 内随机产生粒子的初始位置, 粒子的初始速度在 $[-V_{\max}, V_{\max}]$ 范围内随机产生; 评价初始粒子种群的适应度函数值; 更新种群的个体极值 P_i 与全局极值 P_{gbest}.

Step 3: 判断算法的迭代次数是否大于 $\max K$: 如果大于 $\max K$, 则转 Step 10; 否则, 对于粒子群中的所有粒子, 执行 Step 4.

Step 4: 根据式 (5), (6) 和 4.2 节中粒子速度、位置的更新方法, 更新粒子的位置与速度.

Step 5: 评价每个粒子的适应度函数值.

Step 6: 如果粒子 i 的适应度优于个体极值 P_i 的适应度, 则将 P_i 设置为粒子的当前位置; 如果粒子 i 的适应度优于全局极值 P_{gbest} 的适应度, 则重新设置 P_{gbest} 的索引号.

Step 7: 按照概率 p_c 和 p_m 执行交叉和变异算子.

Step 8: 如果 P_{gbest} 的适应度函数值连续 G 代都相同, 则对当前的粒子种群应用速度扰动策略;

Step 9: 算法的迭代次数增加一次, 返回 Step 3.

Step 10: 输出 P_{gbest} 的解信息, 算法运行结束.

5 仿真实验及结果

5.1 实验参数设置

为了验证改进的粒子群算法 IPSO 的适用性和有效性, 本文将该算法应用于不同规模的 Lot-sizing 问题. 仿真实例的参数设计如下: 时间周期数量 T 取值为 10, 15, 20, 40, 45 和 50; 每个时间周期的平均产品需求量为 100, d_t 在区间 $[50, 150]$ 内随机产生; 单位产品的库存保管费用 h 在区间 $[1, 5]$ 内随机生成; 调整费用 sc 与库存费用 h 有着密切的关系, 在给定各时间周期的平均产品需求量的前提下, 比值 sc/h 与 Lot 的大小有关, 进而影响着各 Lot 之间的时间间隔; 比率 sc/h 分别取值为 50, 100 和 200. 对于每组参数 (T, sc/h), 随机生成 5 个测试实例.

PSO 算法采用如下的参数设置: $c_1 = c_2 = 2$; w 采用线性递减的方法[13], 公式如下:

$$w = w_{\max} - k \times \frac{w_{\max} - w_{\min}}{\max K}. \tag{11}$$

其中: $w_{\max} = 0.9$, $w_{\min} = 0.4$, k 是当前种群的迭代次数, 算法的最大迭代次数 $\max K = 1\,000$; 最大速度 $V_{\max}$ 取值为 4; 交叉概率 $p_c = 0.7$, 变异概率 $p_m = 0.05$, 速度扰动策略中的参数 $G = 20$; 种群规模 Popsize = 20; 针对每个测试实例算法执行 10 次, 取 10 次运算的平均值. 算法采用 C 语言编程实现, 运行环境为一台 PC 机, CPU 为 Pentium IV 2.50 GHz, 操作系统为 Windows XP.

5.2 IPSO 与其他算法性能比较

IPSO 算法的性能从解的质量和算法的求解时间两个方面衡量. 问题的最优解或下界 LB 通过 Cplex 软件计算 MILP 模型 (1)~(4) 获得. 将算法解值 S 与问题的最优解或下界 LB 的相对偏差 gap = $(S-\text{LB})/\text{LB}$ 作为度量标准, 判断算法解质量的优劣. 将本文提出的 IPSO 与 GA[3], DPSO[14] 算法的计算结果进行了比较, 结果如表 1 和表 2 所示. 表中角标 1~3 分别表示 DPSO, GA 和 IPSO 算法, avg_ctime 为 Cplex 的平均计算时间. GA 中交叉、变异概率与本文算法取值相同, 种群规模 GAPopsize = 100; DPSO 与 IPSO 的参数取值相同, 且算法的最大迭代次数都设置为 1 000 代.

表 1 在小规模仿真实例集上本文算法与其他算法的性能比较

instance	T	sc/h	avg_gap^1/%	avg_time^1/s	avg_gap^2/%	avg_time^2/s	avg_gap^3/%	avg_time^3/s	avg_ctime/s
$S_{01} \sim S_{05}$	10	50	0	0.062 4	0	0.193 6	0	0.062 5	1.887 5
$S_{06} \sim S_{10}$	10	100	0	0.062 4	0	0.193 6	0	0.062 6	1.881 2
$S_{11} \sim S_{15}$	10	200	0.113 7	0.062 4	0	0.193 8	0	0.062 4	1.903 1
$S_{16} \sim S_{20}$	15	50	0	0.093 8	0	0.209 4	0	0.093 7	1.906 3
$S_{21} \sim S_{25}$	15	100	0	0.096 8	0	0.218 8	0	0.093 8	1.909 4
$S_{26} \sim S_{30}$	15	200	0	0.1	0	0.215 6	0	0.096 8	1.881 3
$S_{31} \sim S_{35}$	20	50	0	0.125	0	0.224 8	0	0.121 8	1.887 5
$S_{36} \sim S_{40}$	20	100	0	0.131 4	0	0.244 0	0	0.128 0	1.887 5
$S_{41} \sim S_{45}$	20	200	0	0.131 2	0	0.231 0	0	0.128 2	1.900 1

表 2　在大规模仿真实例集上本文算法与其他算法的性能比较

instance	T	sc/h	avg_gap[1]/%	avg_time[1]/s	avg_gap[2]/%	avg_time[2]/s	avg_gap[3]/%	avg_time[3]/s	avg_ctime/s
$S_{46} \sim S_{50}$	40	50	0	0.265 2	0	0.290 6	0	0.250 0	2.900 0
$S_{51} \sim S_{55}$	40	100	0	0.265 6	0.090 4	0.328 2	0	0.256 2	2.928 1
$S_{56} \sim S_{60}$	40	200	0.989 5	0.265 4	0.540 1	0.315 6	0	0.253	2.806 3
$S_{61} \sim S_{65}$	45	50	0	0.290 8	0	0.309 4	0	0.278 2	2.965 6
$S_{66} \sim S_{70}$	45	100	0.115 8	0.299 8	0.307 4	0.350 0	0.071 0	0.293 8	2.987 5
$S_{71} \sim S_{75}$	45	200	0.658 4	0.297 0	0.359 7	0.322 0	0.166 7	0.284 4	2.925 0
$S_{76} \sim S_{80}$	50	50	0	0.325 0	0	0.328 2	0	0.306 2	2.853 2
$S_{81} \sim S_{85}$	50	100	0.043 8	0.331 4	0.030 3	0.371 8	0.026 0	0.325 0	2.887 5
$S_{86} \sim S_{90}$	50	200	0.974 2	0.331 4	0.709 9	0.350 0	0.604 1	0.315 6	2.909 4

从表 1 可以看出, GA 和 IPSO 都得到了问题的最优解, 除了 $S_{11} \sim S_{15}$ 之外, DPSO 也获得了问题的最优解; 从求解时间来看, IPSO 的计算时间与 DPSO 相似, 大约是 GA 的 50%. 此外, 表 2 的测试结果也表明了 IPSO 在解的质量和计算时间两个方面均优于 GA 和 DPSO, 并且算法解值与问题的下界值的相对偏差均低于 1%.

5.3　IPSO 算法性能分析

为了分析交叉、变异算子和速度扰动策略对算法性能的影响, 以参数组 $(T = 20, sc/h = 100)$ 中的一个仿真实例为例, 进行算法性能的测试. 图 3 给出了标准 PSO(SPSO)、带有速度扰动策略的 PSO(IPSO-1)、带有交叉变异算子的 PSO(IPSO-2) 和 IPSO 四个算法的目标函数值变化曲线.

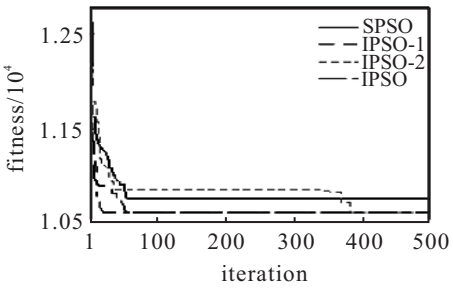

图 3　SPSO, IPSO-1, IPSO-2 和 IPSO 的目标函数值变化曲线

在种群进化早期, IPSO-2 的收敛速度慢于 SPSO, 引入的交叉变异算子扩展了 IPSO-2 中粒子的搜索空间, 使 IPSO-2 在进化后期跳出局部最优点, 并最终收敛到全局最优点. 而 SPSO 缺乏局部最优的逃逸机制, 尤其是在进化后期更容易陷入局部最优位置, 因此未能搜索到全局最优点. IPSO-1 的性能优于 IPSO-2, 说明速度扰动策略通过增加种群的多样性在一定程度上提高了算法的搜索能力. 虽然 IPSO-1 和 IPSO-2 都收敛到全局最优点, 但是从收敛速度来看, IPSO 快于 IPSO-1 和 IPSO-2. 综合上述实验结果, 表明引入的交叉、变异算子和速度扰动策略提高了算法的全局搜索能力, 较好地避免了种群陷入局部最优值.

6　结　论

通过对 Lot-sizing 生产计划问题的描述及分析, 本文提出了改进的 PSO 算法. 在该算法中, 定义粒子的位置为生产的调整状态, 并利用调整费用与库存费用的总费用来衡量个体的优劣. 此外, 该算法利用单切点交叉搜索算子较好地改善了粒子的局部搜索能力, 并引入变异算子和速度扰动策略保证了群体的多样性, 引导算法跳出局部最优值. 随机产生的 90 组仿真实例的实验结果验证了该算法的有效性和适用性.

参考文献(References)

[1] Zhu X, Wilhelm W. Scheduling and lot sizing with sequence-dependent setup: A literature review[J]. IIE Trans, 2006, 38(11): 987-1007.

[2] Jans R, Degraeve Z. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches[J]. European J of Operational Research, 2007, 177(3): 1855-1875.

[3] Xie J, Dong J. Heuristic genetic algorithms for general capacitated Lot-sizing problems[J]. Computers & Mathematics with Applications, 2002, 44(1): 263-276.

[4] Xiao Y, Kaku I, Zhao Q, et al. A variable neighborhood search based approach for uncapacitated multilevel lot-sizing problems[J]. Computers & Industrial Engineering, 2011, 60(2): 218-227.

[5] Han Y, Tang J, Kaku I, et al. Solving uncapacitated multilevel lot-sizing problems using a particle swarm optimization with flexible inertial weight[J]. Computers & Mathematics with Applications, 2009, 57(11/12): 1748-1755.

[6] Dye C Y, Ouyang L Y. A particle swarm optimization for solving joint pricing and lot-sizing problem with fluctuating demand and trade credit financing[J]. Computers & Industrial Engineering, 2011, 60(1): 127-137.

(下转第1091页)