第 27 卷第 10 期
2010 年 10 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 27 No. 10
Oct. 2010

文章编号: 1000−8152(2010)10−1300−07

多级生产批量规划问题的柔性惯量反捕食粒子群算法

韩 毅[1], 蔡建湖[1], 周根贵[1], 李延来[2], 唐加福[2]

(1. 浙江工业大学 经贸管理学院, 浙江 杭州 310023; 2. 东北大学 流程工业综合自动化教育部重点实验室, 辽宁 沈阳 110819)

摘要: 多级生产批量规划(MLLS)是原料需求计划(MRP)中主生产计划(MPS)的关键决策问题, 具有广泛的工业应用; 已被证明是NP-hard类型的组合优化问题. 反捕食粒子群算法(APSO) 是最近提出的一种与粒子群算法(PSO)密切相关的亚启发式算法. 本文提出带柔性惯性权重的反捕食粒子群算法(WAPSO) 对具有指定装配结构而无约束的MLLS问题进行了求解. 本算法对12个小规模benchmark数据集和1个随机产生的较大规模数据进行了测试. 测试结果与遗传算法(GA)和Wagner-Whitin(WW)动态规划算法的结果进行了比较. 结果表明了WAPSO算法的有效性和适用性.

关键词: 多级生产批量规划; 反捕食粒子群算法; 亚启发式算法; 惯性权重; 装配结构

中图分类号: TP29 **文献标识码**: A

Anti-predatory particle-swarm optimization with flexible inertial weight for unconstrained multilevel lot-sizing problems

HAN Yi[1], CAI Jian-hu[1], ZHOU Gen-gui[1], LI Yan-lai[2], TANG Jia-fu[2]

(1. College of Business Administration, Zhejiang University of Technology, Hangzhou Zhejiang 310023, China;
2. Key Laboratory of Integrated Automation of Process Industry, Northeastern University, Shenyang Liaoning 110819, China)

Abstract: Multilevel lot-sizing(MLLS) is a crucial problem in decision-making for the master production scheduling(MPP) of the material requirement plan(MRP), which is with broad industrial applications and has been considered the NP-hard combinatory optimization problem. Anti-predatory particle-swarm optimization(APSO), which is closely related to particle-swarm optimization(PSO), is a recently emerged meta-heuristics. An anti-predatory particle-swarm optimization with flexible inertial weight(WAPSO) is proposed to solve the unconstrained MLLS problem in a given assembly structure. A set of 12 small-sized benchmark data and a randomly generated medium size data are adopted to test the proposed algorithm. The experimental results are compared with those of genetic algorithm(GA) and Wagner-Whitin(WW) dynamic programming algorithm, the results show that WAPSO algorithm is an effective and suitable tool for solving the unconstrained MLLS problem in a given assembly structure.

Key words: multilevel lot-sizing; anti-predatory particle swarm optimization; meta-heuristics; inertial weight; assembly structure

1 引言(Introduction)

物料需求计划(MRP)系统是MRPII/ERP系统[1]的基础, 同时也是供应链管理 SCM(supply chain management)[2,3]的基础. 虽然MRP在企业中是一个十分古老的研究领域, 但是迄今为止它在协调复杂成品的补充决策中仍然起着不可或缺的作用[4]. 多级生产批量计划(MLLS)问题是MRP中主生产计划(MPS)的关键决策问题, 具有广泛的工业背景, 已被证明是NP-hard类型的组合优化问题[5]. 因为MLLS问题的数学模型已经十分成熟, 因此其研究价值不是体现在模型的创新上, 而是体现在寻求更加有效的求解算法上[6]. 在MLLS问题的求解算法方面, 学者们过去经常采用启发式算法或精确算法来求解多级有能力和无能力约束批量计划问题[7]. 近年来, 国内外学者开始尝试采用亚启发式算法求解MLLS问题. Kuik等[8]采用模拟退火SA(simulated annealing)算法对具有装配结构的有能力约束MLLS问题进行了求解; 唐立新等[6]采用GA对装配结构的无能力约束MLLS问题进行了求解; Dellaert等[9,10]采用GA对无能力约束的网络结构MLLS问题进行了求解; 谢金星等[5]采用GA算法对有能力约束的网络结构MLLS问题进行了求解; Jeunet等[4]采用SA算法对无能力约束的网络结构MLLS问题进行了求解; 马慧民等[11]对有能力约束的单级批量计划问

收稿日期: 2008−11−04; 收修改稿日期: 2009−12−15.
基金项目: 国家自然科学基金资助项目(70625001, 70721001, 70671095, 70971017); 浙江省科技计划软科学研究资助项目(2009C35007); 浙江省自然科学基金资助项目(Y1100854); 浙江省社科规划课题资助项目(10CGGL21YBQ).

题采用改进PSO算法进行了求解; 韩毅等[12~14]采用PSO算法、带柔性惯量的PSO算法以及分散搜索算法SS(scatter search)对无资源约束的MLLS问题进行了求解; 韩毅等[15]采用元算法MA(memetic algorithm)对单级有资源约束的MLLS问题进行了求解; Pitakaso等[16]采用蚁群算法AS(ant systems)对无能力约束的网络机构MLLS问题进行了求解.

APSO是最近由Selvakumar[17]提出的一种亚启发式算法, 它是一种加入了反捕食机制的粒子群算法(APSO). 本文基于APSO算法, 通过对APSO的速度更新公式中的惯量ω进行柔性化处理, 形成一种带柔性惯量的APSO算法(WAPSO). 通过对具有装配结构的无资源约束MLLS问题进行了求解, 证明了WAPSO的良好效果.

2　数学模型(Mathematical model)

本文考虑具有装配结构的无能力约束MLLS生产系统, 该系统只包含一种产品, 采用文献[4]的混合整数规划数学模型来描述该系统, 其数学模型为

$$\min C=\sum_{i=1}^{N}\sum_{j=1}^{T}(s_iy_{it}+h_iI_{it}), \tag{1}$$

$$\text{s.t. } I_{it}=I_{it-1}+x_{it}+d_{it},\ \forall i\in N,\ \forall t\in T, \tag{2}$$

$$d_{it}=\begin{cases}D_t, & \text{如果}\Gamma(i)=\varnothing,\\ \sum\limits_{j\in\Gamma(i)}c_{ij}x_{jt}, & \text{其他},\end{cases} \tag{3}$$

$$x_{it}-My_{it}\leqslant 0,\ \forall i\in N,\ \forall t\in T, \tag{4}$$

$$x_{it}\geqslant 0,\ I_{it}\geqslant 0,\ y_{it}\in\{0,1\},\ \forall i\in N,\ \forall t\in T. \tag{5}$$

符号说明如下:

$\Gamma(i)$: 物料i的所有直接后继的集合;

$\Gamma^{-1}(i)$: 物料i的所有直接前驱的集合;

c_{ij}: 生产一个单位物料j所需要物料i的数量;

D_t: 在第t个时段, 外界对产品的需求量;

h_i: 物料i的单位库存费用;

s_i: 物料i的单位生产装配费用;

N: 物料i的总数量;

T: 生产计划期的长度;

M: 非常大的正数.

决策变量如下：

I_{it}: 在第t个时段末, 物料i的总库存量;

d_{it}: 在第t个时段, 物料i的总需求量;

x_{it}: 在第t个时段, 物料i的总生产量;

y_{it}: 二进制变量, 决定在第t个时段是否生产物料i.

如果生产物料i, 则$y_{it}=1$; 否则$y_{it}=0$.

式(1)中的目标函数是在长度为T的计划期间内, 所有物料的装设和库存保管费用的总和. 由于不允许缺货, 每单位物料的生产成本可以忽略不计. 等式(2)是库存平衡方程. 约束(3)表示只有产品才有外部需求, 其他物料的需求由各自物料的直接后继的生产量决定. 约束(4)表示在某个时间段, 如果生产量大于零, 那么在该时间段上装设费用也大于零. 约束(5)强调了不允许延迟交货, 并且各物料的生产数量是非负的.

3　柔性惯量反捕食粒子群算法(WAPSO algorithm)

3.1　粒子群算法(PSO algorithm)

PSO是由Kennedy和Eberhart最先在1995年提出的一种新兴的群智能优化技术. 之后, Shi等[18]引入惯性权重ω来更好地控制开发(exploitation)和探索(exploration), 形成了当前的标准版本. PSO算法是一种基于种群的自适应随机优化技术, 最初用来在连续空间内求解问题. PSO中的每个粒子在3种行为(惯量、认知行为和社会行为)的帮助下搜索全局最优值[19].

基本PSO算法的数学模型如下所示:

速度更新等式:

$$V_{ij}^t=\omega^tV_{ij}^{t-1}+C_1r_1({P_{\text{best}}}_{ij}^{t-1}-X_{ij}^{t-1})+C_2r_2({G_{\text{best}}}_i^{t-1}-X_{ij}^{t-1}). \tag{6}$$

位置更新等式:

$$X_{ij}^t=X_{ij}^{t-1}+V_{ij}^t, \tag{7}$$

i表示粒子的维数, j表示某个粒子, t表示当前迭代数.

3.2　反捕食粒子群算法[17](APSO algorithm)

在鸟群捕食的过程中, 认知和社会行为帮助鸟群在捕食过程中定位食物(最好解). 但是鸟群在捕食的同时也要躲避它们的天敌们, 也就是说鸟群不仅有捕食的行为同时也存在反捕食的现象. APSO算法就是基于上述思想的一种PSO算法, APSO将捕食者们作为最差位置引入到PSO的模型中. PSO算法的认知部分被分割成两个部分, 一部分是好的经验, 一部分是坏的经验. 前者是粒子对自身经历的最好位置的记忆, 后者是粒子对自身经历的最差位置(捕食者的位置)的记忆. 同样地, APSO算法将PSO算法的社会行为部分也被分割成全局最好经验和全局最差经验部分. 由此, APSO算法的数学模型中速度更新等式如下所示:

$$V_{ij}^t=\omega^tV_{ij}^{t-1}+C_1r_1({P_{\text{best}}}_{ij}^{t-1}-X_{ij}^{t-1})+$$

$$C_2r_2(X_{ij}^{t-1}-P_{\mathrm{worst}_{ij}}^{t-1})+$$
$$C_3r_3(G_{\mathrm{best}_i}^{t-1}-X_{ij}^{t-1})+$$
$$C_4r_4(X_{ij}^{t-1}-G_{\mathrm{worst}_i}^{t-1}). \tag{8}$$

等式(8)中粒子个体所经历的最好位置对粒子的影响用$C_1r_1(P_{\mathrm{best}_{ij}}^{t-1}-X_{ij}^{t-1})$表示, 粒子个体的最差位置对粒子的影响用$C_2r_2(X_{ij}^{t-1}-P_{\mathrm{worst}_{ij}}^{t-1})$表示, $C_3r_3(G_{\mathrm{best}_i}^{t-1}-X_{ij}^{t-1})$是群体的全局最好位置对粒子的影响, $C_4r_4(X_{ij}^{t-1}-G_{\mathrm{worst}_i}^{t-1})$是群体的全局最差位置对粒子的影响. 位置更新仍然根据等式(7)来进行.

在粒子的行为中加入坏的经验可以增加粒子的额外探索能力, 粒子总是能够绕过这些最差的位置并尽量占据更好的位置[17]. 文献[17]采用APSO算法对具有连续变量的经济分配问题进行了求解, 取得了不错的效果.

3.3 WAPSO算法及运算规则定义[12,13](WAPSO algorithm and definitions for operations)

WAPSO算法是引入了柔性惯量概念的APSO算法, 与APSO算法相比, WAPSO能够更好地控制速度的长度, 使速度的长度既能够增加又能够减少, 降低了不同迭代之间出现相同速度的概率. 本文要求解问题的变量是离散的, 因此需要对WAPSO的速度更新公式(8)中的速度和位置更新公式(7)中的一些运算规则进行重新定义[12,13]. 在WAPSO算法的执行过程中, 如果出现不可行解, 则立即将不可行解修改为可行解[4].

1) 位置Y: 位置也就是问题的编码, 是由N行和T列的决策矩阵组成, 矩阵中的每个元素都是一个二进制0–1变量, 比如某个位置$Y_k=[(y_{11},\cdots,y_{1T})\ \cdots\ (y_{N1},\cdots,y_{NT})]$.

2) 搜索空间S: 搜索空间是所有可能的决策矩阵的集合$S=Y_k$.

3) 速度V: 速度是由数对集合构成的, 速度的长度就是集合中数对的个数, 比如一个速度$V=((i_q,j_q))$, 这里$\|V\|$是速度的长度, $i\in\{1,\cdots,N\}$, $j\in\{1,\cdots,T\}, q\in\{1,\cdots,\|V\|\}$.

4) 速度“加”速度($\oplus$): 速度相“加”定义为一个速度和另一个速度进行并集操作. 假设$V_1=((1,4),(2,3),(5,7))$及$V_2=((1,7),(2,3))$, 则$V_1\oplus V_2=((1,4),(2,3),(5,7),(1,7))$.

5) 速度“减”速度($\ominus$): 速度相“减”定义为一个速度和另一个速度进行差并操作. 所谓差并, 就是首先去掉两个速度中相同的数对, 然后将剩余部分进行合并. 假设$V_1=((1,4),(2,3),(5,7))$及$V_2=((1,7),(2,3))$, 则$V_1\ominus V_2=((1,4),(5,7),(1,7))$.

6) 位置“加”速度($\boxplus$): 位置与速度的相“加”就是将位置中的对应位按照速度中的每个数对依次进行变异操作. 在变异的过程中有可能出现不可行解, 这时就要根据文献[9]提到的累积变异方法对不可行解进行可行化处理. 假设一个包含1种产品和3种物料的装配结构生产系统, 该系统生产决策矩阵(位置)用Y表示, 如果一个速度$V=((1,4),(2,2))$, 则$Y\boxplus V$的结果如图1所示, 其中: (a)表示将(1,4)位进行变异, (b)表示将(2,2)位进行变异.

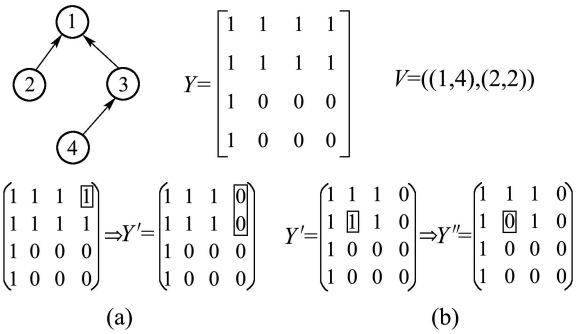

图 1 位置与速度的“加”运算

Fig. 1 “Plus” operation between a position and a velocity

在将Y中的(1,4)位置点进行变异时, 不仅是位置点(1,4)发生了变化, 位置点(2,4)也随之进行了调整. 这是因为这一步的变异操作后, Y成为不可行解. 根据累积变异规则, 需要同时修改其他位置点以保证Y是可行解.

7) 位置“减”位置($\boxminus$): 一个位置与另一个位置相“减”的结果是一个速度, 是通过记录两个位置的不同之处而得到的. 图2给出两个位置相“减”的操作过程.

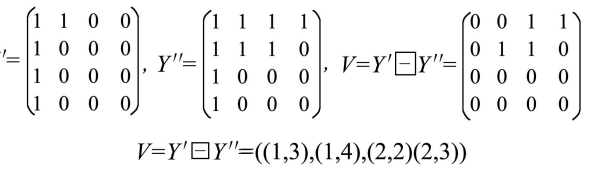

图 2 位置与位置的“减”运算

Fig. 2 “Minus” operation between two positions

8) 系数“乘”速度($\otimes$): 一个系数C与速度V相“乘”的结果根据如下两种情况来定:

a) 如果$|C|<1$, 设小于或等于$|C|*\|V\|$的最大整数为$\|CV\|$, 则$C\otimes V=((i_q,j_q))$, $q\in\{1,\cdots,\|V\|\}$. 例如: 如果$C=0.6$, $V=((2,4),(2,5),(4,3))$, 则$\|V\|=3$, $|C|*\|V\|=1.8$, $\|CV\|=1$, $C\otimes V=V'=((2,4))$.

b) 如果$|C|\geqslant 1$, 则$C\otimes V$的结果还是V本身, 这样可以避免重复数对的出现.

如果惯性权重为正实数, 则WAPSO算法按等式(9)和(10)来执行; 如果惯性为负实数, 则WAPSO算法按等式(11)和(10)来执行. 等式(9)(10)和(11)中经过重新定义的运算符间的优先关系与数学运算中运算符之间的优先关系相同.

$$V_{ij}^t = (\omega^t \otimes V_{ij}^{t-1}) \oplus \{[(C_1 r_1) \otimes (P_{\mathrm{best}_{ij}}^{t-1} \boxminus X_{ij}^{t-1})] \oplus [(C_2 r_2) \otimes (X_{ij}^{t-1} \boxminus P_{\mathrm{worst}_{ij}}^{t-1})] \oplus [(C_3 r_3) \otimes (G_{\mathrm{best}_i}^{t-1} \boxminus X_{ij}^{t-1})] \oplus [(C_4 r_4) \otimes (X_{ij}^{t-1} \boxminus G_{\mathrm{worst}_i}^{t-1})]\}, \tag{9}$$

$$X_{ij}^t = X_{ij}^{t-1} \boxplus V_{ij}^t, \tag{10}$$

$$V_{ij}^t = (|\omega^t| \otimes V_{ij}^{t-1}) \ominus \{[(C_1 r_1) \otimes (P_{\mathrm{best}_{ij}}^{t-1} \boxminus X_{ij}^{t-1})] \ominus [(C_2 r_2) \otimes (X_{ij}^{t-1} \boxminus P_{\mathrm{worst}_{ij}}^{t-1})] \ominus [(C_3 r_3) \otimes (G_{\mathrm{best}_i}^{t-1} \boxminus X_{ij}^{t-1})] \ominus [(C_4 r_4) \otimes (X_{ij}^{t-1} \boxminus G_{\mathrm{worst}_i}^{t-1})]\}. \tag{11}$$

算法执行步骤:

Step 1　随机产生初始种群P, 随机产生每个粒子j的初始速度V_j, 初始化所有相关参数信息.

Step 2　根据每个粒子j的当前位置Y_j计算适应值F_j, 保留个体最优位置P_{best}和个体最差位置P_{worst}, 根据各粒子的个体最优位置, 找出全局最优位置G_{best}; 根据各粒子的个体最差位置, 得到全局最差位置G_{worst}.

Step 3　根据公式$\omega = W \times [(\omega_{\mathrm{init}} - 0.4) \times (\mathrm{Max_Iter} - \mathrm{Cur_Iter})/\mathrm{Max_Iter}] + 0.4$计算当前的惯性权重$\omega$的值. 这里$\omega_{\mathrm{init}} = 0.9$, Max_Iter是算法的最大迭代次数, Cur_Iter是算法当前的迭代次数, W是一个随机产生的值为-1或1的变量.

Step 4　如果$\omega \geqslant 0$, 则按等式(9)和(10)更新每个粒子j的速度V_j和位置Y_j; 如果$\omega < 0$, 则按等式(11)和(10)更新每个粒子j的速度V_j和位置Y_j.

Step 5　计算各个粒子的适应值F_i, 更新个体最优位置P_{best}、个体最差位置P_{worst}、全局最优位置G_{best}以及全局最差位置G_{worst}.

Step 6　检查是否达到最大迭代次数或是否满足其他停止准则. 如果满足停止标准, 输出G_{best}和最优值F_{best}; 否则转到Step 3.

4　计算实验(Experimental results)

4.1　实验参数(Parameters for experiments)

实验部分采用文献[9]中的12个小规模问题以及1个随机产生的较大规模问题对本文WAPSO算法的有效性和适用性进行了测试, 并将求解结果与APSO算法、GA算法和WW算法的求解结果进行了比较. 对于不同规模的问题, 实验参数有所不同. 算法的参数中, 一部分来自于文献, 一部分是经过前期的测试和结果比较而得出的. 对于小规模问题和较大规模问题, WAPSO算法和APSO算法的参数保持不变: 种群大小为30; C_1, C_2, C_3, C_4分别为1.6, 0.4, 1.8, 0.2[17]; r_1, r_2, r_3, r_4是取值为 0 到 1 之间的随机实数; 算法最大迭代次数为100; 速度的最大长度为$0.02 * N * T$. 算法停止准则为: 如果在20次连续迭代内最优值没有明显变化, 则停止程序. 关于小规模问题, GA算法的种群大小为30, 交叉、变异和交换操作的概率为0.5,0.4和0.1, 最大迭代次数为100; 对于较大规模问题, GA算法的种群大小为60, 最大迭代次数为500. 算法停止准则是: 如果在200次连续迭代内最优值没有明显变化, 则停止程序. 本文所有算法采用MS.Visual Basic6. 0进行编程实现, 并且对于每一组算例独立运行50次.

4.2　小规模问题的实验结果比较(Comparison results for small-sized instances)

表 1 给出了 4 种算法对小规模问题的求解结果, 图3显示了WAPSO算法、APSO算法和GA算法在50次迭代内对问题S-2求解后的平均值收敛曲线图.

从表 1 中可以看到, 在计算时间相差不大的情况下, WAPSO算法的达优性能及平均值都要好于 APSO 算法和 GA 算法, 同时从标准偏差来看, WAPSO算法也体现出了较好的抗干扰性. WAPSO算法的性能之所以能够优于APSO算法, 很大程度上是因为WAPSO算法中不同迭代之间出现相同速度的概率较低. 图3的平均值收敛曲线可以更直观地体现出WAPSO算法的收敛性比其他算法的性能更好.

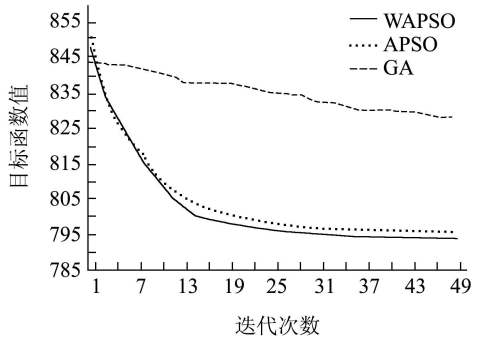

图 3　3种算法的平均值收敛曲线

Fig. 3　Convergence curves of mean values from three algorithms for small-sized instances

表 1 4种算法的小规模问题求解结果

Table 1 Comparison results of 4 algorithms for small-sized instances

问题	最优解	问题	最好解	最差解	平均值	平均计算时间/s	达优率/%	标准方差
S-1	984.5	WAPSO	984.5	1061.5	1008.2	1.63	28	22.24
		APSO	984.5	1060	1014.5	1.63	24	23.93
		GA	984.5	1106.5	1030.67	1.55	14	36.59
		WW	1097	1097	1097	—	0	0
S-2	785.2	WAPSO	785.2	812.65	794.09	1.71	38	9.23
		APSO	785.2	833.9	796.22	1.75	28	11.05
		GA	785.2	905.5	816.51	1.54	2	28.78
		WW	893	893	893	—	0	0
S-3	580.6	WAPSO	580.6	624.93	585.73	1.45	44	9.14
		APSO	580.6	620.2	586.36	1.55	28	7.49
		GA	581.03	628.87	592.56	1.53	0	10.63
		WW	692.75	692.75	692.75	—	0	0
S-4	1115	WAPSO	1115	1153.75	1126.93	1.89	20	9.91
		APSO	1120	1185	1138.1	1.95	0	14.7
		GA	1115	1270	1156.73	1.55	6	34.79
		WW	1135	1135	1135	—	0	0
S-5	944.75	WAPSO	944.75	992.25	962.2	2.27	4	10.77
		APSO	959.75	1083.75	990.25	2.29	0	25.91
		GA	949.75	1100.75	1014.27	1.55	0	35.57
		WW	977.25	977.25	977.25	—	0	0
S-6	759.4	WAPSO	759.4	812.65	774.58	1.9	6	10.11
		APSO	761.4	836.9	783.33	2.26	0	15.58
		GA	773.9	917.7	817.39	1.58	0	34.84
		WW	791.9	791.9	791.9	—	0	0
S-7	1024.5	WAPSO	1024.5	1143.75	1054.21	1.63	44	32.3
		APSO	1024.5	1147.75	1071.38	1.66	16	32.84
		GA	1024.5	1241.5	1096.22	1.55	8	53.27
		WW	1146.5	1146.5	1146.5	—	0	0
S-8	879.6	WAPSO	879.6	996	913.93	1.59	32	37.29
		APSO	879.6	1008	926.35	1.54	28	42.58
		GA	879.6	1106.15	953.9	1.53	8	55.48
		WW	1043.2	1043.2	1043.2	—	0	0
S-9	724.62	WAPSO	724.62	860.83	755.21	1.41	18	31.37
		APSO	724.62	844.05	758.47	1.52	8	27.47
		GA	724.62	894.94	795.07	1.55	4	44.42
		WW	947.33	947.33	947.33	—	0	0
S-10	1250	WAPSO	1250	1311.25	1269.23	1.68	16	13.41
		APSO	1250	1303.75	1272.18	1.74	16	15.49
		GA	1250	1415	1301.75	1.49	2	39.14
		WW	1315	1315	1315	—	0	0
S-11	1269	WAPSO	1269	1335.25	1286.34	1.88	24	16.73
		APSO	1269	1344	1293.46	2.02	8	17.33
		GA	1269	1430	1333.64	1.54	2	33.87
		WW	1313.75	1313.75	1313.75	—	0	0
S-12	1318	WAPSO	1318	1380.5	1341.51	1.78	8	17.37
		APSO	1318	1378.6	1343.02	1.95	2	15.75
		GA	1323.6	1491.9	1387.41	1.57	0	35.52
		WW	1392.2	1392.2	1392.2	—	0	0

注: 表1中“—”表示计算时间很少.

4.3 较大规模问题的实验结果比较(Comparison results for a medium-sized instance)

图4给出了较大规模问题的产品结构图, 表2给出了4种算法对较大规模问题的求解结果, 图5显示了WAPSO算法和APSO算法在50次迭代内的平均值收敛曲线图.

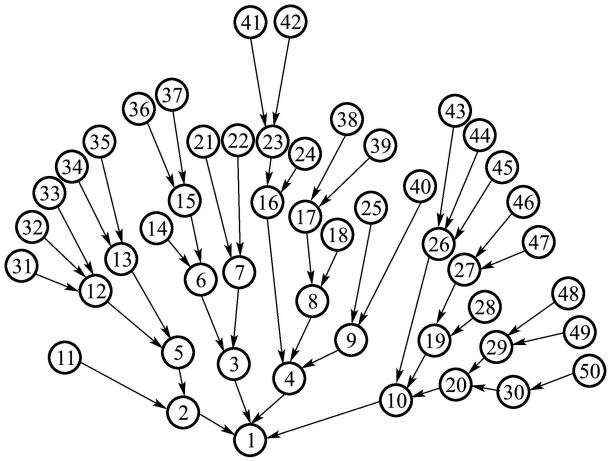

图 4　较大规模问题的产品结构图

Fig. 4　Bill of materials for a medium-sized instance

表 2　较大规模问题的求解结果比较

Table 2　Comparison results of four algorithms for a medium-sized instance

算法	最好解	最差解	平均值	标准方差	平均时间/s
WAPSO	4921	19102	7517.2	6632.57	5.2
APSO	4921	33541	8545.7	4086.97	7.1
GA	4921	25037	9241.9	5012.88	33
WW	13207	13207	13207	0	0.1

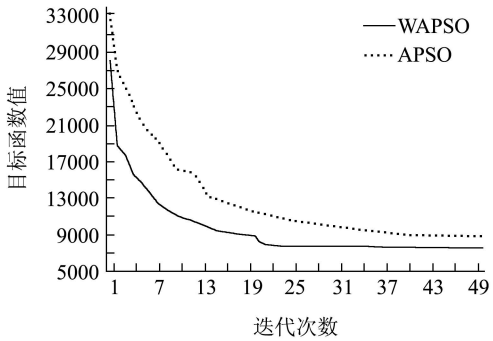

图 5　2种算法的平均值收敛曲线图

Fig. 5　Convergence curves of mean values from two algorithms for a medium-sized instance

图5比图3更明显的反映出WAPSO算法的良好的收敛性. 从表2中可看到WAPSO算法的平均值好于APSO算法和GA算法, 虽然其标准差较差, 但其求解时间相对较少. 如果让WAPSO算法和APSO算法运行与GA算法相同的计算时间, WAPSO算法的达优率为80%, 平均值为5017.08, 标准方差为327.05, 最差解为6581; APSO算法的达优率为66%, 平均值为5250.1, 标准方差为672.62, 最差解为7331.

5 结论(Conclusion)

MLLS问题是机械制造加工企业中每天都要面对的问题, 至今仍没有有效的最优解算法. 本文扩展了WAPSO算法的应用领域、改进了APSO算法并且将处理连续优化问题的APSO算法进行离散化处理, 形成可以求解离散优化问题的APSO算法和WAPSO算法. 对具有装配结构的MLLS问题进行了求解, 通过计算实验结果表明: WAPSO算法是一种有效的和适用的求解工具, 对生产实践活动起了积极的指导作用.

参考文献(References):

[1] 陈伯成, 叶伟雄, 李英杰. 虚企业与ERP软件关系研究[J]. 管理科学学报, 2002, 5(5): 62 – 72.
(CHEN Bocheng, YE Weixiong, LI Yingjie. Research on relationship between “virtual enterprise” and ERP software[J]. *Journal of Management Sciences in China*, 2002, 5(5): 62 – 72.)

[2] 刘丽文. 供应链管理思想及其理论和方法的发展过程[J]. 管理科学学报, 2003, 6(2): 81 – 88.
(LIU Liwen. Survey on evolution of SCM theory and methods[J]. *Journal of Management Sciences in China*, 2003, 6(2): 81 – 88.)

[3] 陈安, 刘鲁. 供应链管理问题的研究现状与挑战[J]. 系统工程学报, 2000, 15(2): 179 – 186.
(CHEN An, LIU Lu. An overview of supply chain management: an operational perspective[J]. *Journal of Systems Engineering*, 2000, 15(2): 179 – 186.)

[4] JEUNET J, JONARD N. Single-point stochastic search algorithms for the multi-level lot-sizing problem[J]. *Computers and Operations Research*, 2005, 32(4): 985 – 1006.

[5] XIE J, DONG J. Heuristic genetic algorithms for general capacitated lot-sizing problems[J]. *Computers and Mathematics with Applications*, 2002, 44(1/2): 263 – 276.

[6] 唐立新. CIMS下生产批量计划理论及其应用[M]. 北京: 科学出版社, 1999.
(TANG Lixin. *Theory and Application of Lot-Sizing Problems in CIMS*[M]. Beijing: Science Press, 1999.)

[7] 谢金星, 姜启源, 邢文训, 等. 能力受限的批量问题的数学模型与算法新进展[J]. 运筹学杂志, 1996, 15(1): 1 – 12.
(XIE Jinxing, JIANG Qiyuan, XING Wenxun, et al. New advances in models and algorithms of constrained lot-sizing problems[J]. *OR Transactions*, 1996, 15(1): 1 – 12.)

[8] KUIK R, SALOMON M. Multi-level lot-sizing problem: evaluation of a simulated annealing heuristic[J]. *European Journal of Operations Research*, 1990, 45(1): 25 – 37.

[9] DELLAERT N, JEUNET J. Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm[J]. *International Journal of Production Economics*, 2000, 38(5): 1083 – 1099.

[10] DELLAERT N, JEUNET J, JONARD N. A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs[J]. *International Journal of Production Research*, 2000, 68(3): 241 – 257.

[11] 马慧民, 叶春明, 柳毅. 基于改进粒子群算法的生产批量计划问题研究[J]. 计算机集成制造系统, 2006, 12(9): 1417 – 1500.
(MA Huimin, YE Chunming, LIU Yi. Improved particle swarm optimization algorithm for dynamic lot-sizing problem[J]. *Computer Integrated Manufacturing Systems*, 2006, 12(9): 1417 – 1500.)

[12] 韩毅, 唐加福, 牟立峰, 等. 粒子群算法求解无能力约束生产批量计划问题[J]. 管理科学学报, 2008, 11(5): 33 – 40.
(HAN Yi, TANG Jiafu, MU Lifeng, et al. Particle swarm optimization algorithm for solving uncapacitated multilevel lot-sizing problems[J]. *Journal of Management Sciences in China*, 2008, 11(5): 33 – 40.)

[13] 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 and Mathematics with Applications*, 2009, 57(11/12): 1748 – 1755.

[14] 韩毅, 卢永虎, 周根贵, 等. Scatter search算法求解无能力约束多级生产批量计划问题[J]. 系统仿真学报, 2009, 13(21): 3902 – 3905.
(HAN Yi, LU Yonghu, ZHOU Gengui, et al. Scatter search algorithm for unconstrained multilevel lot-sizing problems[J]. *Journal of System Simulation*, 2009, 13(21): 3902 – 3905.)

[15] 韩毅, 唐加福, 王立岩, 等. 单级有资源约束的生产批量计划问题的元算法[J]. 东北大学学报(自然科学版), 2009, 30(8): 1111 – 1114.
(HAN Yi, TANG Jiafu, WANG Liyan, et al. A memetic algorithm for single-level capacitated lot-sizing problems[J]. *Journal of Northeastern University (Natural Science)*, 2009, 30(8): 1111 – 1114.)

[16] PITAKASO R, ALMEDER C, DOERNER K, et al. A MAX-MIN ant system for unconstrained multi-level lot-sizing problems[J]. *Computers and Operations Research*, 2007, 34(9): 2533 – 2552.

[17] SELVAKUMAR A, THANUSHKODI K. Anti-predatory particle swarm optimization: solution to nonconvex economic dispatch problems[J]. *Electric Power Systems Research*, 2008, 78(1): 2 – 10.

[18] SHI Y, EBERHART R. A modified particle swarm optimizer[C] //*Proceedings of the IEEE International Conference on Evolutionary Computation*. New York: IEEE, 1998: 69 – 73.

[19] 贾兆红, 陈华平, 孙耀晖. 混合粒子群算法在柔性工作车间调度中的应用[J]. 系统仿真学报, 2007, 19(20): 4743 – 4747.
(JIA Zhaohong, CHEN Huaping, SUN Yaohui. Hybrid particle swarm optimization for flexible job-shop scheduling[J]. *Journal of System Simulation*, 2007, 19(20): 4743 – 4747.)

作者简介:

韩 毅 (1979—), 男, 博士, 讲师, 主要从事生产计划与物流运作优化、智能优化算法及应用等方面的研究, E-mail: hanyi@zjut.edu. cn, 本文通讯作者;

蔡建湖 (1977—), 男, 博士, 副教授, 主要从事生产与物流运作管理方面的研究;

周根贵 (1958—), 男, 博士, 教授, 主要从事管理信息系统方面的研究;

李延来 (1971—), 男, 在东北大学流程工业综合自动化教育部重点实验室作博士后研究工作, 主要从事工程产品优化设计与管理、供应链与物流运作管理等的研究;

唐加福 (1965—), 男, 博士, 教授, 博士生导师, 目前研究方向为供应链管理、进化算法、Fuzzy优化理论与方法, E-mail: jftang@mail.neu.edu.cn.