第 28 卷第 2 期
2011 年 2 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 28 No. 2
Feb. 2011

文章编号: 1000−8152(2011)02−0215−08

多阶段多产品调度问题的链式智能体遗传算法

吴亚丽, 张万良, 张立香

(西安理工大学 自动化与信息工程学院, 陕西 西安 710048)

摘要: 将遗传算法的编码方式与智能体系统的演化结构相结合, 提出一种求解多阶段多产品调度问题的链式智能体遗传算法. 算法采用基于订单序列的编码方式, 采用一种新的后向指派规则实现编码和可行调度间的一一对应. 通过各智能体与其邻域环境的竞争与合作以及自身的自学习操作实现种群的演化过程. 对多阶段多产品调度问题的仿真结果表明: 链式智能体遗传算法与新的后向指派规则相结合, 不仅增加了种群多样性, 而且提高了算法的收敛性能, 是求解多阶段多产品调度问题的有效算法.

关键词: 多阶段多产品调度问题; 启发式规则; 遗传算法; 智能体系统

中图分类号: TP391　　**文献标识码:** A

Chain-like agent genetic algorithm for multi-stage multi-product scheduling problem

WU Ya-li, ZHANG Wan-liang, ZHANG Li-xiang

(Automation and Information Engineering School, Xi'an University of Technology, Xi'an Shaanxi 710048, China)

Abstract: Combined the coding characteristic of the genetic algorithm with the evolution structure in the multi-agent system, a chain-like agent genetic algorithm is proposed to solve the multi-stage multi-product scheduling problem. The order-sequences-based encoding means is adopted, and the one-to-one correspondence between the encoding and feasible scheduling is achieved by new post-assignment rules. The population evolution is implemented by the operators of agent such as competition and cooperation with the dynamic neighboring environment and self-learning operator with its own knowledge. The simulation results of multi-stage multi-product scheduling problem show that the combination of chain-like agent genetic algorithm with the new heuristic rule not only increases the diversity of the population but also improves the convergent performance. It is effective in solving the multi-stage multi-product scheduling problem.

Key words: multi-stage multi-product scheduling problem; heuristic rule; genetic algorithm; agent system

1　引言(Introduction)

多阶段多产品调度问题(multi-stage multi-product scheduling problem, MMSP)广泛存在于化工、钢铁、制药等流程工业中, 是一类典型的NP完全问题. 其特征是各阶段均存在并行加工单元, 目标是确定所有订单的排序及各加工单元的分配以使某性能指标达到最优. 遗传算法作为一种基于种群搜索的全局概率搜索优化算法, 对优化函数本身既不要求连续也不要求可微, 具有很强的鲁棒性和适应性, 在复杂调度问题的求解上得到了广泛应用. 文献[1]提出了多阶段混合调度问题的混合整数规划模型, 并采用遗传算法进行求解. 由于简单遗传算法收敛速度慢, 且易出现早熟现象, 因此提出了多种改进遗传算法; 文献[2]提出一种新的遗传算法编码方法求解多阶段混合调度问题, 算法设计了相应的交叉和变异操作算子, 能够保证个体的合法性, 同时又具有遗传算法要求的随机性; 文献[3]以当代种群平均适应度为期望Ex, 根据云模型"3En"规则确定熵En, 由X条件云发生器自适应调整交叉变异概率, 提出云自适应遗传算法; 文献[4]把简化的二次插值法融入实数编码遗传算法, 构成适于求解全局优化问题的混合遗传算法; 文献[5]将遗传算法与启发式规则相结合, 采用活动调度技术, 通过对大规模MMSP问题进行求解, 得到了在不同目标下的调度; 文献[6]提出了一种基于多种群思想的遗传算法, 设计了适合连续离散规划问题求解的交叉与变异算子以及评价函数参数. 上述改进算法在一定程度上提高了算法的收敛性能和探索性能.

近年来, 基于智能体的计算已应用于计算机学科的各个领域. 多智能体系统能适应各种环境变化, 能够不断自我完善和发展, 是一种理想的调度模型[7]. 将遗传算法灵活的编码方式和与智能体系统

收稿日期: 2009−07−03; 收修改稿日期: 2010−02−25.
基金项目: 陕西省自然科学基金资助项目(2010JQ8006); 陕西省教育厅科学研究计划资助项目(2010JK711).

的求解能力相结合更能充分发挥求解NP-hard问题的巨大潜力. 文献[8]提出的组合优化多智能体算法对于解决各种类型的欺骗函数有着良好的性能; 文献[9]将智能体对环境的感知和反作用的能力与遗传算法的搜索方式相结合, 提出了一种多智能体的遗传算法(multi-agent genetic algorithm, MAGA); 梁昌勇[10]等将均匀设计的试验设计方法引入多智能体遗传算法中, 提高了算法的优化性能; 文献[11]针对John Holland反转算子在数值优化中的不合理性, 提出了具有轮盘反转算子的多智能体遗传算法; 文献[12, 13]对智能体的邻域环境进行改进, 提出了一维智能体遗传算法, 减少了次优个体过早取得“顶端优势”而导致过早收敛的发生.

本文针对大规模MMSP调度问题的特点, 将遗传算法的编码方式与链式智能体系统结构和演化优势相结合, 基于新的后向指派规则提出一种求解MMSP问题的链式智能体遗传算法(chain-like agent genetic algorithm, CAGA). 文中针对目标函数采用新的后向指派策略将订单序列指派到相应的加工单元, 从而实现编码和可行解的对应关系; 演化过程采用链式智能体的网络结构, 通过竞争算子、交叉算子和自学习算子实现动态邻域竞争选择、交叉和变异, 较好地保持了种群多样性, 加快了算法的收敛速度. 通过对大规模MMSP问题的仿真实验, 结果证明了算法和规则相结合的正确性和有效性.

2 MMSP问题描述(Description of MMSP)

多阶段多产品问题描述如下: 假设在批过程中, 总共有$M_{\rm T}$个加工单元, 需要加工N个订单, 每个订单对应一类产品, 每个订单都有一个预先确定的交货期d_j, 且必须经过M个加工阶段. 每个加工阶段均有多个并行加工单元来加工所有订单. 在每个阶段中, 一个订单只能选取一个单元进行加工, 且阶段中所有并行加工单元均只在当前阶段有效, 不同阶段的加工单元具有不同的属性.

在整个加工过程中, 每个加工单元都存在订单转换时间, 用来表示在加工不同订单的过程中, 加工单元需要的准备时间. 有些订单禁止在一些单元上加工, 某些订单之间禁止转换, 禁止转换和禁止加工在问题中称为CP约束. 调度问题的目标通常采用基于时间的目标函数. 本文采用的总的加工时间和总的拖延时间目标如下:

$$\min PT = \sum_{j=1}^{N}(d_j - H_j), \tag{1}$$

$$\min T = \sum_{j=1}^{N} T_j, \tag{2}$$

其中: $T_j = \max\{C_j - d_j, 0\}$为订单$j$的拖延时间, C_j为订单j的完成时间, d_j为订单j的交货期, H_j为订单j的开始时间.

MMSP的数学模型见文献[14]. 由于MMSP是一个高度复杂的组合问题, 其模型涉及的变量和约束条件非常多, 对于小规模问题, 用MILP方法可以在较短的时间获得最优解, 而随着问题规模的增大, 模型变量和约束条件指数增长, MILP方法很难在可接受的时间内求得最优解. 因此, 多采用智能优化算法和启发式规则相结合来求解大规模MMSP问题.

3 MMSP问题的链式智能体遗传算法(Chain-like agent genetic algorithm for MMSP)

3.1 Agent编码以及种群产生(Agent coding and population generating)

单个智能体采用订单序列来表示种群的个体, 即$P = (P_1, P_2, \cdots, P_S)$, 其中$P_k = (P_{k1}, P_{k2}, \cdots, P_{kN})(k = 1, 2, \cdots, S)$为阶段数, P_{ki}为P_k中第$i(i = 1, 2, \cdots, N)$个订单, 且$P_{ki} \in \{1, 2, \cdots, N\}$. 图1表示了一个$5 \times 10(S = 5, N = 10)$的个体.

阶段1	阶段2	阶段3	阶段4	阶段5
19265847310	65738291104	10394857261	74859310261	93716210548

图 1 $5 \times 10(S = 5, N = 10)$的个体的编码结构

Fig. 1 Coding structure of chromosome with 5 stages and 10 orders

对于一个具有N个订单和S个阶段的MMSP问题其种群的产生为: 从第1个阶段开始, 对每个阶段, 随机产生1到N的N个不同的随机数, 作为该阶段的订单排序, 直到S个阶段的订单排序均产生完, 则得到1个个体. 以此方法产生PopSize个个体, 将得到具有PopSize个个体的种群.

3.2 Agent结构及其邻域环境(Structure and neighbor environment of agent)

定义 1 Agent结构及能量.

每个智能体数据信息结构表示为: a =(body, place, local), 其中: body为一候选解亦即其被优化问题所决定的适应值; place为智能体a在环境中的位置; local为智能体a的邻域$\text{local}(a) = (n_{\rm pre}, n_{\rm next})$, 其中$n_{\rm pre}$和$n_{\rm next}$分别表示智能体$a$的前一个智能体和后一个智能体; $n_c = (\text{place}(c), \text{trust}(c)), c = \text{pre}, \text{next}$; place($c$)和trust($c$)分别表示智能体$c$所处的位置以及可信度.

智能体的能量定义为: $E(a) = 1/f(a)$且$a \in S'$, 其中: S'为问题搜索空间, $f(a)$为最小化问题的目标函数值, 也即遗传算法中的适应度函数. 可以看出, 能量越大, 适应值越小.

定义 2 环境.

常用的多智能体遗传算法中的环境采用格子环境, 在进化过程中容易产生次优个体过早取得“顶端优势”而导致过早收敛的发生. 因此本文采用链式智能体环境结构, 如图2所示.

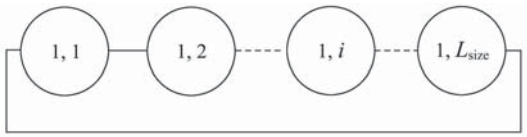


图 2 链式智能体网络结构

Fig. 2 Chain-like network structure of agent

每个智能体都生存在链式环境L中, 称L为智能体链, 其规模为$1 \times L_{\text{size}}$(L_{size}为整数). 由于智能体只有局部感知能力, 故它只能与邻域内的几个智能体相互作用.

定义 3　Agent邻域.

智能体邻域的概念采用文献[12]中的定义. 即: 假设$L_{1,i}$是格子坐标为$(1,i)$的智能体, $i = 1, 2, \cdots, L_{\text{size}}$, $L_{1,i}$的初始邻域$N_{1,i}$被定义为$N_{1,i} = \{L_{1,i_1}, L_{1,i_2}\}$, 其中:

$$i_1 = \begin{cases} i-1, & i \neq 1, \\ L_{\text{size}}, & i = 1, \end{cases} \tag{3}$$

$$i_2 = \begin{cases} i+1, & i \neq L_{\text{size}}, \\ 1, & i = L_{\text{size}}. \end{cases} \tag{4}$$

由上述定义可以看出, 与多智能体遗传算法的网格环境相比, 链式智能体的邻域个体相对较少, 在一定程度上降低了算法的探索能力, 但也减少了算法陷入局部极值的几率, 同时计算量明显降低.

3.3 Agent解码及能量计算(Agent decoding and energy calculating)

在MMSP问题中, 由于每个阶段都存在多个并行加工单元, 而且各单元的加工属性根据订单的不同而不同. 因此, 将一个给定的订单序列, 按照不同的方法指派到并行加工单元, 产生的调度结果不同. 启发式规则的优劣直接影响算法的收敛速度和寻优能力. 文献[5]根据加工时间和处理时间之和最小, 提出了一种启发式规则, 但该规则只是对订单加工时间和转换时间做了优化, 没有优化同一个订单中两个相邻加工过程的空闲时间. 对此本文提出了一个新的启发式规则: 确保每个订单在各阶段的实际完成时间尽可能接近其在该阶段的预期完成时间; 尽可能减少各单元的闲置时间. 规则实现如下:

设当前要指派的订单为$P_{ki}(i = 1, 2, \cdots, N)$, 要指派的阶段为$k(k = 1, 2, \cdots, S)$, 单元为$m(m = 1, 2, \cdots, M_k)$, M_k为阶段k中并行单元数量. 令$j = P_{ki}$, 则有订单P_{ki}在阶段k中预期完成时间为$d_{j,k}$, 则启发式规则具体实现过程如下:

Step 1　若$i = 1$, 则将订单P_{ki}指派给阶段k中加工订单P_{ki}所用时间最小的加工单元. 否则转到Step 2;

Step 2　令阶段k中的单元$m = 1$;

Step 3　找出单元m的所有有效闲置时间, 从中选出与$d_{j,k}$最靠近的有效闲置时间, 将订单P_{ki}插入到该时间上且使得插入后的完成时间尽可能接近$d_{j,k}$, 将插入订单后得到的完成时间保存在一个数组中;

Step 4　若$m = M_k$, 则转到Step 5, 否则令$m = m + 1$, 转到Step 3;

Step 5　从Step 3中得到的数组中找出最大的数, 将订单P_{ki}指派到与该数所对应的加工单元.

利用文献[5]中的规则和本文规则, 在相同约束下对图1所示个体产生的调度甘特图分别为图3中(a)和(b). 从图中可以看出, 本文的启发式规则具有明显的优势, 极大地减少了调度中的闲置时间.

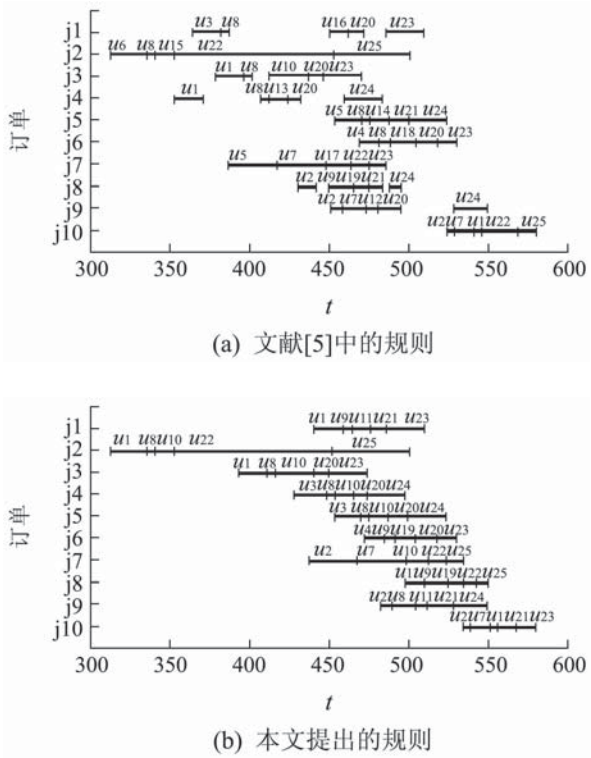


图 3 不同规则下调度序列的甘特图

Fig. 3 Gantt chart of scheduling sequences based on different rules

在本文的启发式指派规则下, 其可行调度及能量的计算步骤为:

Step 1　取出当前个体P, 并令阶段$k = S$, $m = 1$, $i = 1$;

Step 2　从P_k中取出P_{ki}, 令$j = P_{ki}$;

Step 3　如果$k = S$, 则令$d_{j,k} = d_j$, 否则令$d_{j,k} = H_j^{k+1}$, H_j^{k+1}为订单j在$k+1$阶段的开始时间;

Step 4 根据本文提出的启发式规则, 将P_{ki}指派到阶段k中的加工单元m上;

Step 5 令$H_j^k = d_{j,k} - (x_{lj} + p_{jkm})$, 其中: x_{lj}为订单1到订单j的转换时间, p_{jkm}为订单j在阶段k中m单元中的加工时间;

Step 6 若$i = N$, 转到Step 7, 否则令$i = i + 1$, 转到Step 2;

Step 7 若k不等于1, 令$k = k - 1$, $i = 1$转到Step 2, 否则转到Step 8;

Step 8 令$H_j = H_j^1$, 计算目标值$f(P) = \sum_{j=1}^{N}(d_j - H_j)$, 最终得到个体$P$的调度.

Step 9 计算智能体能量$E(P) = 1/f(P)$.

3.4 Agent进化算子(Agent evolution operator)

针对大规模多阶段批调度的特点, 设计了竞争算子、交叉算子和自学习算子来实现智能体行为, 其中竞争算子和交叉算子是利用邻域信息来实现智能体之间的竞争与合作, 自学习算子利用智能体自身的知识来增加能量.

竞争算子 假设在格子点位置$(1, i)$的智能体$L_{1,i} = (l_{i,1}, l_{i,2}, \cdots, l_{i,n})$, $L_{1,i}$邻域内能量最大的智能体为$M_{1,i}$, 且$M_{1,i} = (m_{i,1}, m_{i,2}, \cdots, m_{i,n})$, 其中$n$为变量的维数, 且$n = S \times N$. 即: $M_{1,i} \in \text{Neibors}_{1,i}$, $\forall a \in \text{Neibors}_{1,i}, E(a) \leqslant E(M_{1,i})$都成立. 若$L_{1,i}$满足式$E(L_{1,i}) > E(M_{1,i})$, 则可继续存活在智能体网络中, 否则必须死亡, 空出的格点将被新的智能体代替. 新智能体的产生算法为:

输入: 邻域内能量最大的智能体

$$M_{1,i} = (m_{i,1}, m_{i,2}, \cdots, m_{i,n});$$

输出: 执行竞争之后产生新的智能体$C_{1,i}$, 且

$$C_{1,i} = (c_{i,1}, c_{i,2}, \cdots, c_{i,n});$$

Step 1 初始化$C_{1,i}$, 将$M_{1,i}$赋给$C_{1,i}$;

Step 2 在$(1, S)$之间随机选择一个整数k, 在$(1, N)$之间随机选择2个整数$i, j (i \neq j)$, 将$C_{1,i}$第k段中的第i个和第j个数之间的数做逆序排列;

Step 3 如果$E(C_{1,i}) > E(L_{1,i})$, 计算相应的适应值, 转到Step 4, 否则结束;

Step 4 更新智能体网格与智能体能量, 结束.

交叉算子 智能体以概率P_{c}与其邻域内的智能体进行交叉操作. 假设a', b'分别为a和b通过交叉产生的子代. 如果$E(a') > E(a)$, 以a'来替代a. 否则, 保持原有的a不变. 同样, 若$E(b') > E(b)$, 以b'来替代b. 否则, 保持原有的b不变. 交叉算子的实现过程为:

输入: 参与交叉的智能体a, b并且$a = (a_1, a_2, \cdots, a_n)$, $b = (b_1, b_2, \cdots, b_n)$, 交叉概率为$P_{\text{c}}$;

输出: 执行交叉之后产生的新智能体为a', b';

Step 1 初始化a', b', 令$a' = a, b' = b, k = 1$;

Step 2 在$(1, N)$之间随机选择两个整数$i, j (i \neq j)$, 将a', b'第k段中第i个和第j个数之间的数按部分匹配交叉[5]操作;

Step 3 若k不等于S, 则令$k = k+1$, 转到Step 2, 否则转到Step 4;

Step 4 分别计算a', b'的适配值;

Step 5 更新智能体网格与智能体能量, 结束.

自学习算子 仅有竞争和交叉算子会导致智能体系统的多样性变差, 因此, 设计了智能体的自学习行为来实现局部搜索, 同时增加整个种群的多样性[15]. 算法描述如下:

输入: 进行自学习操作的智能体a, 自学习概率为P_{m};

输出: 执行自学习之后产生的新智能体为c;

Step 1 初始化c, 将a赋给c, 令$k = 1$;

Step 2 在$(1, N)$之间随机选择2个整数$i, j (i \neq j)$, 将c第k段中第i个和第j个数之间的数做逆序排列;

Step 3 若k不等于S, 则令$k = k+1$, 转到Step 2, 否则转到Step 4;

Step 4 计算c的适配值;

Step 5 更新智能体网格与智能体能量, 结束.

3.5 求解MMSP问题的链式智能体遗传算法实现步骤(Chain-like agent genetic algorithm for solving MMSP)

设L_i表示第i代智能体种群, local_i为第i代智能体链的局部生存环境, L_i^1和L_i^2为L_i和L_{i+1}的2个中间种群, B_i表示$\{L_0, L_1, \cdots, L_i\}$中能量最高的智能体, B_i^1是L_i中能量最高的智能体, P_{c}和P_{m}分别为执行邻域交叉操作和自学习操作所需的概率, $U(0, 1)$为$(0, 1)$之间的随机数. 采用一维智能体遗传算法求解MMSP问题的具体步骤如下:

Step 1 选定智能体链长度L_{size}, 最大迭代次数MAX, 初始化智能体链L_0, 局部生存环境local_0, 更新B_0, 令$i \leftarrow 0$.

Step 2 对智能体链中的任意一个智能体执行动态邻域竞争算子, 得到L_i^1;

Step 3 对L_i^1上的每个智能体, 如果$U(0, 1) < P_{\text{c}}$, 则执行交叉操作, 得到L_i^2, 同时更新智能体的局部生存环境local_i;

Step 4 对L_i^2中的智能体, 如果$U(0, 1) < P_{\text{m}}$, 则执行自学习操作, 得到L_{i+1};

Step 5 从L_{i+1}找出能量最高的智能体B_{i+1}^1, 比较B_{i+1}^1和B_i, 如果$E(B_{i+1}^1) > E(B_i)$, 则将B_{i+1}^1给

B_{i+1}, 否则将B_i分别给B_{i+1}和B_{i+1}^1;

Step 6　判断是否满足终止条件, 若满足则停止, 并输出B_i, 否则令$i \leftarrow i+1$转到Step 2.

3.6 算法参数选择(Parameters of Dynamic chain-like agent genetic algorithm)

算法中需要选择的参数主要有编码串长度L、种群规模PopSize、交叉概率$P_{\rm c}$、变异概率$P_{\rm m}$以及停止准则.

编码串长度　根据订单特点使用十进制编码, 个体的编码结构长度为$L = N \times S$.

种群规模　为便于和文献资料进行对比, 同时顾及运行时间因素, 本文针对问题的小、中和大规模分别取群体数为200, 300和600, 确保小规模问题算法的CPU时间不超过10 s, 中规模问题不超过1 min, 而大规模问题则不超过5 min.

交叉概率和变异概率　本文采用自适应交叉概率和变异概率, $P_{\rm c}$和$P_{\rm m}$能够随适应度自动改变[16]. $P_{\rm c}$和$P_{\rm m}$计算如下:

$$P_{\rm c} = \begin{cases} k_1 \sin(\dfrac{\pi}{2}\dfrac{f_{\max} - f'}{f_{\max} - f_{\rm avg}}), & f' > f_{\rm avg}, \\ k_2, & f' \leqslant f_{\rm avg}, \end{cases} \quad (5)$$

$$P_{\rm m} = \begin{cases} k_3 \sin(\dfrac{\pi}{2}\dfrac{f_{\max} - f}{f_{\max} - f_{\rm avg}}), & f > f_{\rm avg}, \\ k_4, & f \leqslant f_{\rm avg}. \end{cases} \quad (6)$$

式中: $k_1 = 1.0$, $k_2 = 1.0$, $k_3 = 0.5$, $k_4 = 0.5$; $f_{\max}$是群体中最大的适应度; $f_{\rm avg}$是每代群体的平均适应度; f'是要交叉的2个个体中较大的适应度; f是要变异的个体的适应度值.

停止准则　本文采用的停止准则为: 连续20代个体平均适应度的差异小于0.01.

4 实例仿真与结果分析(Simulation and analysis)

为验证本文启发式规则和链式智能体算法的有效性, 本文选择了两个多阶段多产品调度问题进行仿真实验.

4.1 具有资源约束的多阶段多产品调度问题仿真及结果分析(Simulation and analysis for MMSP with resource constraints)

具有资源约束的多阶段多产品调度问题分为5个阶段, 包括12个订单、12个加工单元. 其中各阶段的单元分配、具体加工处理时间、订单交货期、单元开始时间、订单之间的转换时间及各种资源约束见参考文献[17].

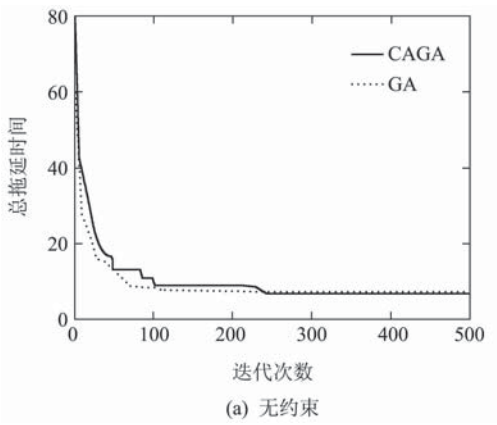
为了验证链式智能体遗传算法的有效性, 算法以式(2)为目标函数, 分别考虑4类约束: I)没有资源约束、II)只有人力资源约束、III)具有人力和电力资源约束、IV)全约束(人力、电力和流量约束). 通过MILP, GA和CAGA等3种方法对不同约束问题进行求解比较, 3种算法的各种性能统计结果(包括最好值Best、最差值Worst、平均值Mean、方差SD以及CPU时间)见表1.

表 1 资源约束问题的仿真结果

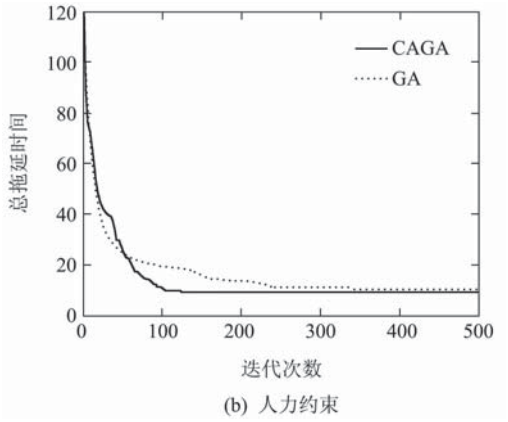
Table 1 Results of experiment for MMSP with resource constraints

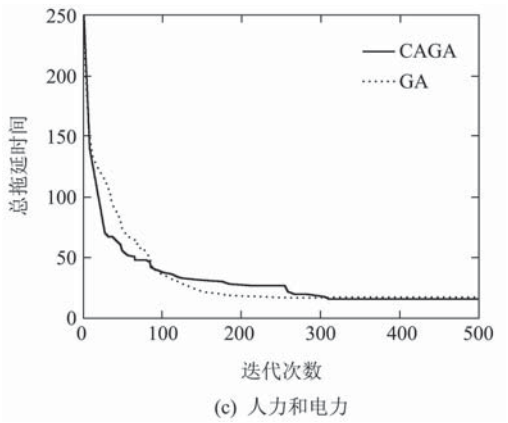
资源约束	MILP[17]		GA					CAGA				
	Best	CPU/s	Worst	Mean	Best	SD	CPU/s	Worst	Mean	Best	SD	CPU/s
I	9.3	69.73	7.79	7.18	7.15	0.03	13	6.91	6.64	6.31	0.03	15
II	11.9	70.41	10.7	9.79	9.02	0.24	17	9.8	9.5	9.12	0.04	18
III	18.3	119.3	18.69	17.87	17.08	0.23	26	18.1	16.2	14.2	1.29	32
IV	31.6	114	30.45	29.85	28.03	0.73	28	33.6	27.5	26.3	3.78	29

从统计结果可以看出, 与MILP算法相比, 遗传算法和链式智能体遗传算法在求解速度和求解效率上都有很大程度的提高. CAGA在搜索解的质量上要优于普通遗传算法. 但由于CAGA是对遗传算法的一种改进, 链式智能体遗传算法的网络结构比普通遗传算法复杂, 所以其搜索速度稍慢于遗传算法, 然而换来了更好的寻优性能. 表中数据还表明, CAGA算法得到的最优值、最差值、平均值均好于GA算法, 进一步表明CAGA算法具有更好的搜索性能. 图4(a)～(d)给出了4种资源约束下GA和CAGA算法各运行30次后的平均值演化曲线.



(a) 无约束





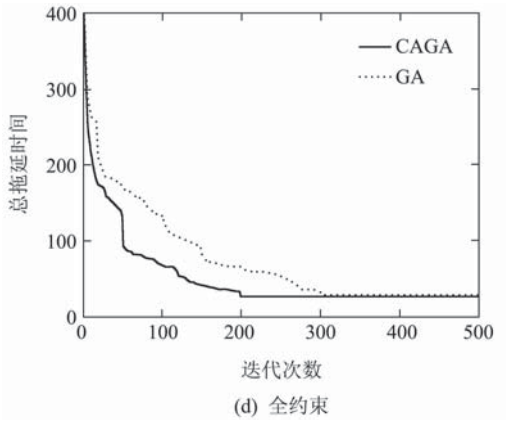


图 4 不同资源约束下调度问题平均值演化曲线

Fig. 4 Average curve of MMSP with different resource constraints

从曲线可以看出, 链式智能体遗传算法的搜索速度总体来说要快于普通遗传算法, 尤其是在考虑人力约束和考虑全约束时. 这表明链式网络结构的加入提高了算法的局部搜索能力和全局的收敛速度.

4.2 大规模多阶段多产品调度问题的仿真实例及结果分析(Simulation and analysis for large-scale MMSP)

为了进一步验证CAGA在大规模MMSP问题上的有效性以及本文提出的后向指派规则的适用性, 本文采用具有5个阶段、24个订单及25个加工单元的大规模MMSP问题, 其中各阶段的单元分配、具体加工处理时间、订单交货期及订单之间的转换时间见参考文献[5].

对上述问题, 以式(1)为求解目标, 通过MILP, OR+GA(文献[5]中的启发式规则和遗传算法相结合), NR+GA(本文规则和文献[5]中遗传算法相结合), NR+CAGA(基于新规则的动态链式智能体遗传算法)4种方法对不同规模问题进行求解比较. 程序均独立运行30次, 各种性能指标(包括最好值Best、最差值Worst、平均值Mean、方差SD以及CPU时间)的统计结果如表2所示.

从表2中的数据可以看出: GA算法和CAGA算法的搜索性能远优于MILP算法, 这表明智能优化算法在求解大规模调度问题时有比数学规划方法更好的性能; 基于新规则的两种遗传算法求得的结果均比文献中提供的解小, 说明新规则更利于对优化目标的搜索, 更能提高算法的寻优能力; 在新规则下, 随着订单数的增加, CAGA算法得到的最好解、最差解、平均解以及方差均比GA算法好, 表明CAGA算法在求解大规模问题时具有更好的搜索性能, 但随之带来的问题是计算时间稍长, 说明链式结构的引入增加了算法的复杂程度. 在求解5, 8和10个订单问题时, 两种算法在30次运行时均找到了最好解.

表 2 大规模多阶段多产品调度问题的仿真结果

Table 2 Results of experiment for large-scale MMSP

Order	PopSize	MILP[5]		OR+GA[5]		NR+GA					NR+CAGA				
		Best	CPU/s	Best	CPU/s	Worst	Mean	Best	SD	CPU/s	Worst	Mean	Best	SD	CPU/s
5	200	499.4	0.43	499.4	1	499.4	499.4	499.4	0	1	499.4	499.4	499.4	0	1
8	200	736.9	99.56	732.9	1	730.9	730.9	730.9	0	5	730.9	730.9	730.9	0	9
10	300	909.3	119.5	876.4	5	863.9	863.9	863.9	0	11	863.9	863.9	863.9	0	19
12	300	1184.7	125.02	1038.9	6	1036.9	1030.8	1022.4	22.19	20	1031.9	1028.6	1022.4	12.16	24
16	600	2048.6	202.5	1788.8	14	1791.2	1781	1774.3	32.34	40	1762.3	1759.3	1751.3	7.91	88
20	600	2982.5	227.17	2456.8	26	2410.3	2402	2399.5	18.46	87	2402.2	2391.4	2377.1	114.21	144
24	600	3912	374.72	3239.3	40	3066.7	3050.3	3036.5	90.9	132	3059.8	3046.8	3030.4	120.32	218

图5(a)~(f)给出了订单数分别为8, 10, 12, 16, 20, 24等6种情形下采用本文启发式规则时两种算法各运行30次后的平均值演化曲线. 从曲线中可以看出:

与文献中的算法相比, 两种算法的最终收敛结果均得到了比文献更优的解, 新规则下两者均具有很好的收敛速度和寻优性能; 在同样规则下, CAGA算法比GA的收敛速度更快、寻优能力更强, 链式智能体遗传算法总是能找到更优的结果, 而普通遗传算法即使改变规则也很难克服易出现早熟收敛的现象, 但引入链式智能体系统的结构后, CAGA算法能更好的保持种群多样性, 从而避免过早收敛的情况.

综上, 新的启发式规则的使用, 提高了整个算法的收敛速度和寻优能力, 大幅减小了不必要的搜索空间. 而引入智能体系统结构得到的CAGA算法, 能更好的保持种群多样性, 避免过早收敛的情况发生, 获得了更好的寻优结果.

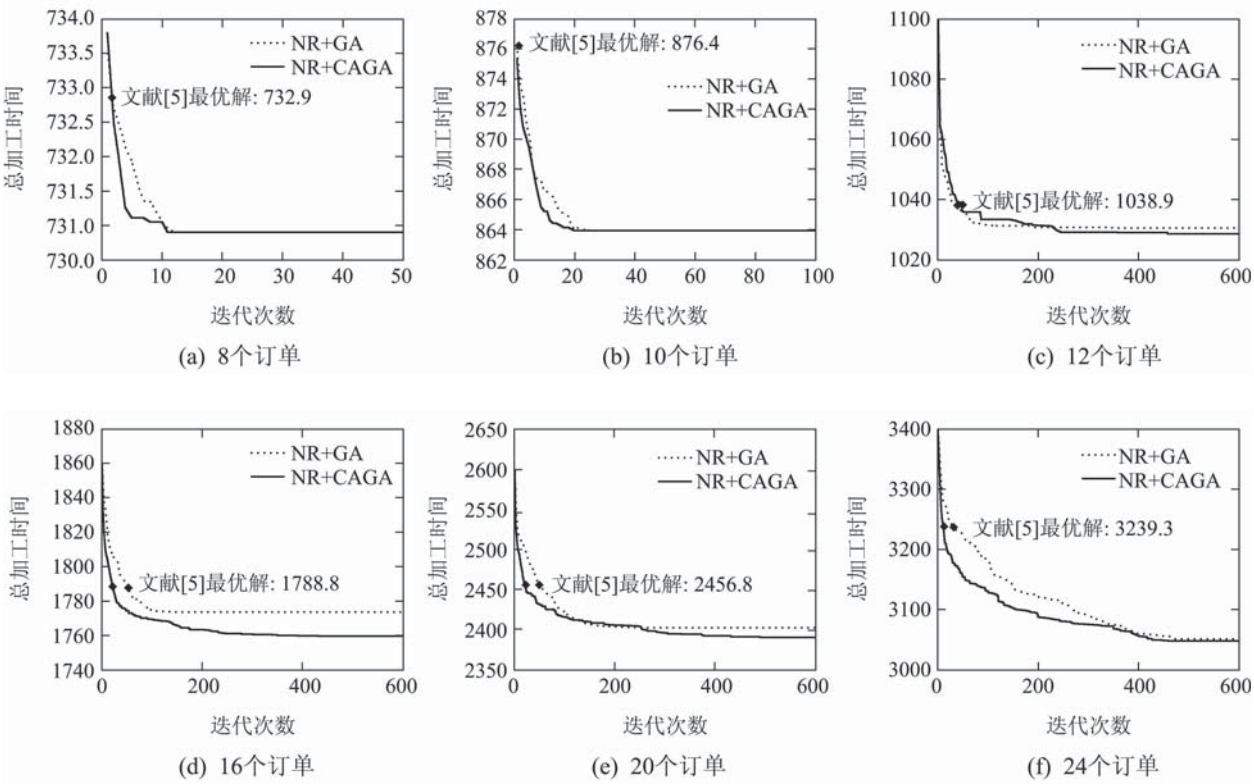


图 5　不同规模MMSP问题的平均值演化曲线

Fig. 5　Average curve of MMSP with different scale

5　结论(Conclusion)

本文针对MMSP的特点, 结合目标函数提出了一种基于新的后向指派启发式规则, 并将该规则与遗传算法的编码方式、智能体系统的演化结构相结合, 构造了一种求解多阶段多产品调度问题的链式智能体遗传算法. 算法中种群的单个智能体代表了问题的一个可行解, 采用了基于订单序列的遗传算法十进制编码方法, 根据后向指派策略将订单序列指派到相应的加工单元, 完成了编码与可行调度的一一对应关系. 种群的演化通过智能体与其邻域环境的竞争与合作以及自身的自学习操作来实现. 链式结构的引入更好地保持了种群多样性, 加快了算法收敛速度. 通过大规模多阶段多产品调度问题的仿真, 结果表明: 链式智能体遗传算法不仅增加了种群多样性, 而且加快了算法的收敛速度; 该算法与新的后向指派启发式规则的结合使用, 更是大大地提高了大规模多阶段多产品调度问题的收敛速度和寻优能力, 是求解复杂调度问题的一种有效算法.

参考文献(References):

[1] 庞哈利, 郑秉霖. 多阶段混合Flow Shop调度问题及其遗传求解算法[J]. 控制与决策, 1999, 14(S1): 565 – 568.
(PANG Hali, ZHENG Binglin. Genetic algorithms for multistage hybrid flow shop scheduling problem[J]. *Control and Decision*, 1999, 14(S1): 565 – 568.)

[2] 王万良, 姚明海, 吴云高, 等. 基于遗传算法的混合Flow-shop调度方法[J]. 系统仿真学报, 2002, 14(7): 863 – 869.
(WANG Wanliang, YAO Minghai, WU Yungao, et al. Hybrid flow-shop scheduling approach based on genetic algorithm[J]. *Journal of System Simulation*, 2002, 14(7): 863 – 869.)

[3] 戴朝华, 朱云芳, 陈维荣. 云自适应遗传算法[J]. 控制理论与应用, 2007, 24(4): 646 – 650.

(DAI Zhaohua, ZHU Yunfang, CHEN Weirong. Adaptive genetic algorithm based on cloud theory[J]. *Control Theory & Applications*, 2007, 24(4): 646 – 650.)

[4] 李宏, 焦永昌, 张莉, 等. 一种求解全局优化问题的新混合遗传算法[J]. 控制理论与应用, 2007, 24(3): 343 – 348.
(LI Hong, JIAO Yongchang, ZHANG Li, et al. Novel hybrid genetic algorithm for global optimization problems[J]. *Control Theory & Applications*, 2007, 24(3): 343 – 348.)

[5] HE Y H, CHI W H. Genetic algorithm for large-size multi-stage batch plant scheduling[J]. *Chemical Engineering Science*, 2007, 62(5): 1504 – 1523.

[6] 姚志红, 赵国文, 韩冰. 多种群变换遗传算法及其在优化调度中的应用[J]. 控制理论与应用, 2001, 18(6): 882 – 886.
(YAO Zhihong, ZHAO Guowen, HAN Bing. The comparison of multi-reproduction groups of genetic algorithms and its application in the optimization schedule[J]. *Control Theory & Applications*, 2001, 18(6): 882 – 886.)

[7] 张磊, 张瑞林. 基于多智能体的动态车间调度系统[J]. 计算机系统应用, 2009, (3): 35 – 41.
(ZHANG Lei, ZHANG Ruilin. Dynamic job shop scheduling system based on multi-agent[J]. *Computer Systems & Applications*, 2009, (3): 35 – 41.)

[8] 钟伟才, 刘静, 刘芳, 等. 组合优化多智能体进化算法[J]. 计算机学报, 2004, 27(10): 1341 – 1353.
(ZHONG Weicai, LIU Jing, LIU Fang, et al. Combinatorial optimization using multi-agent evolutionary algorithm[J]. *Chinese Journal of Compuers*, 2004, 27(10): 1341 – 1353.)

[9] ZHONG W C, LIU J, XUE M Z, et al. A multi-agent genetic algorithm for global numerical optimization[J]. *IEEE Transactions on Systems, Man, and Cybernetics–part B: Cybernetics(S1083-4419)*, 2004, 34(2): 1128 – 1141.

[10] 梁昌勇, 陆青, 张恩桥, 等. 基于均匀设计的多智能体遗传算法研究[J]. 系统工程学报, 2009, 24(1): 109 – 113.
(LIANG Changyong, LU Qing, ZHANG Enqiao, et al. Research on multi-agent genetic algorithm based on uniform design[J]. *Journal of Systems Engineering*, 2009, 24(1): 109 – 113.)

[11] 张俊岭, 梁昌勇, 杨善林. 具有轮盘反转算子的多agent算法用于线性系统逼近[J]. 控制理论与应用, 2009, 26(1): 39 – 45.
(ZHANG Junling, LIANG Changyong, YANG Shanlin. Effective multi-Agent algorithm with roulette inversion operator for approximating linear systems[J]. *Control Theory & Applications*, 2009, 26(1): 39 – 45.)

[12] 李勇明, 周峨, 曾孝平. 一种可用于数值优化的一维智能体遗传算法的研究[J]. 系统仿真学报, 2009, 21(1): 23 – 27.
(LI Yongming, ZHOU Di, ZENG Xiaoping. Research of one dimensional agent genetic algorithm for global numerical optimization[J]. *Journal of System Simulation*, 2009, 21(1): 23 – 27.)

[13] ZENG X P, LI Y M, JIAN Q. A dynamic chain-like agent genetic algorithm for global numerical optimization and feature selection[J]. *Neurocomputing*, 2009, 72(4/6): 1214 – 1228.

[14] CHI W H, AVANEESH G. A novel MILP formulation for short-term scheduling of multistage multi-product batch plants[J]. *Computers & Chemical Engineering*, 2000, 24(2): 1611 – 1617.

[15] 潘晓英, 焦李成. 项目优化调度的多智能体社会进化算法[J]. 计算机研究与发展, 2008, 45(6): 998 – 1003.
(PAN Xiaoying, JIAO Licheng. A multi-agent social evolutionary algorithm for project optimization scheduling[J]. *Journal of Computer Research and Development*, 2008, 45(6): 998 – 1003.)

[16] 王万良, 吴启迪. 生产调度智能算法及应用[M]. 北京: 科学出版社, 2007.
(WANG Wanliang, WU Qidi. *Intelligent Algorithm of Production Scheduling and Its Application*[M]. Beijing: Science Press, 2007)

[17] PABLO A MARCHETTI, JAIME CERDA. A general resource-constrained scheduling framework for multistage batch facilities with sequence-dependent changeovers[J]. *Computers and Chemical Engineering*, 2009, 33(4): 871 – 886.

作者简介:

吴亚丽 (1975—), 女, 副教授, 主要研究方向为复杂系统建模、优化与仿真、Petri网的理论与应用研究, E-mail: yliwu@xaut.edu.cn;

张万良 (1980—), 男, 硕士研究生, 主要研究方向为大系统理论与方法, E-mail: zhwal@163.com;

张立香 (1984—), 女, 硕士研究生, 主要研究方向为系统优化与调度, E-mail: zhlx0820@163.com.