文章编号: 1001-0920 (2016) 10-1772-07 DOI: 10.13195/j.kzyjc.2015.1309

基于GMOGSO的多目标流水车间调度问题

徐震浩, 李继明, 顾幸生

(华东理工大学 信息科学与工程学院，上海 200237)

摘 要: 针对缓冲区有限的多目标流水车间调度问题, 提出一种基于Pareto最优的广义多目标萤火虫算法. 通过引入交换子和交换序将基本萤火虫算法离散化, 并将算法拓展为全局搜索过程和局部搜索过程. 进化初期采用全局搜索将种群推向较优区域, 进化中后期采用捕食搜索策略使算法主体在全局搜索和局部搜索间智能切换, 从而保证全局与局部的平衡. 动态变步长策略进一步增强了算法搜索能力. 通过算例测试验证了所提出算法的有效性.

关键词: 有限缓冲区；萤火虫算法；多目标优化；捕食搜索

中图分类号: TP301 **文献标志码:** A

Multi-objective flow shop scheduling problem based on GMOGSO

XU Zhen-hao, *LI Ji-ming*, *GU Xing-sheng*

(School of Information Science and Technology，East China University of Science and Technology，Shanghai 200237, China．Correspondent：XU Zhen-hao，E-mail：xuzhenhao@ecust.edu.cn)

Abstract: To deal with the multi-objective flow shop scheduling with limited buffer, a Pareto-based general multi-objective glowworm swarm optimization(GMOGSO) algorithm is proposed. The concepts of the swap operator and swap sequence are introduced to make the continuous GSO discrete. To balance the convergence speed and accuracy, the GSO algorithm is developed into two different processes including the global optimization one and local optimization one. The global optimization process is used to improve the quality of the initial population, then the predatory search strategy is used to coordinate local and global exploration. The method of variant step length enhances the exploratory capability. The proposed GMOGSO algorithm is compared to other algorithms, and the results show the effectiveness of the proposed algorithm.

Keywords: limited buffers；glowworm swarm optimization；multi-objective optimization；predator search algorithm

0 引 言

传统流水车间调度问题(FSP)假定设备间缓冲区存储能力无限. 在实际加工中, 由于缓冲区空间和生产工艺的限制, 缓冲区大小通常有限. 近年来, 有限缓冲区流水车间调度问题受到了越来越多的关注[1-2]. Smutnicki[3]研究了针对两台机器上的有限缓冲区流水车间调度问题; Liu 等[4]针对有限缓冲区置换流水车间调度问题提出了混合蚁群算法; 胡蓉等[5]采用混合差分进化算法求解随机有限缓冲区流水车间调度问题; Qian 等[6]以最小化最大完成时间和最大拖期时间为目标, 提出了混合差分进化算法. 实际生产中, 决策者往往需要考虑多方面因素, 如加工时间和能耗等, 在当前能源日趋紧张的环境下, 研究面向节能的生产调度问题符合实际生产需求. 萤火虫优化算法(GSO)是一种新兴仿生算法, 近年来已逐渐成为研究热点[7-8]. 如, Krishnanand 等[9]将变步长机制引入萤火虫算法, 使得步长随着迭代次数增加而相应减小; Zhou 等[10]引入了多种群的思想, 并进一步提出了部落的概念, 将种群分为两个层次分别进行优化; 曾冰等[11]重新定义了萤火虫算法的相关操作, 将其离散化, 并用于求解装配序列规划问题.

本文对基本GSO进行扩展, 提出一种基于Pareto最优的广义萤火虫多目标优化算法(GMOGSO), 并将其应用于有限缓冲区多目标流水车间调度问题.

1 问题描述

有限缓冲区流水车间调度可以描述如下: n 个产品依次在 m 台设备上加工, 同一时刻一个产品只能在一台设备上加工, 并且一台设备只能加工一个产品, 各设备上 n 个产品的加工顺序相同. 相邻两设备 j 与 $j-1$ 之间存在大小为 $B_{j-1,j}$ 的缓冲区, 若产品 i 在设

收稿日期: 2015-10-25；**修回日期:** 2016-01-20.

基金项目: 国家自然科学基金项目(61104178, 61174040).

作者简介: 徐震浩(1976−), 女, 副教授, 博士, 从事生产计划与调度的研究；顾幸生(1960−), 男, 教授, 博士生导师, 从事过程建模、生产计划与调度等研究.

备 $j-1$ 上加工完成后恰好加工下一工序的设备 j 正在使用, 则产品 i 进入缓冲区; 若缓冲区已满, 则工件阻塞在当前设备上.

令 $\pi=(k_1,k_2,\cdots,k_n)$ 为所有工件的一个排序, S_{ij} 为产品 i 在设备 j 上的开始加工时间, t_{ij} 为其加工时间, $C(k_i,j)$ 为产品 k_i 在设备 j 上的完工时间, 则有

$$S_{k_1,1}=0; \tag{1}$$

$$S_{k_1,j}=S_{k_1,j-1}+t_{k_1,j-1},\ j=2,3,\cdots,m; \tag{2}$$

$$S_{k_i,1}=\begin{cases} S_{k_{i-1},1}+t_{k_{i-1},1},\ i=2,3,\cdots,B_{j,j+1}+1; \\ \max\{S_{k_{i-1},1}+t_{k_{i-1},1},S_{i-B_{12}-1,2}\}, \\ \qquad i>B_{j,j+1}+1; \end{cases} \tag{3}$$

$$S_{k_i,j}=\begin{cases} \max\{S_{k_i,j-1}+t_{k_i,j-1},S_{k_{i-1},j}+t_{k_{i-1},j}\}, \\ \quad i=2,3,\cdots,B_{j,j+1}+1,\ j=2,3,\cdots,m; \\ \max\{S_{k_i,j-1}+t_{k_i,j-1},S_{k_{i-1},j}+t_{k_{i-1},j}, \\ \quad S_{i-B_{j,j+1}-1,j+1}\},\ i>B_{j,j+1}+1; \end{cases} \tag{4}$$

$$C(k_i,j)=S_{k_i,j}+t_{k_i,j},$$

$$i=1,2,\cdots,n,\ j=1,2,\cdots,m; \tag{5}$$

$$E_{\rm pe}=\sum_{j=1}^{m}\sum_{i=1}^{n}E_{{\rm pe}_{ij}}=\sum_{j=1}^{m}\sum_{i=1}^{n}\partial_j\times t_{ij}; \tag{6}$$

$$E_{\rm ae}=\sum_{k=1}^{m}\sum_{i,j=1}^{n}E_{{\rm ae}_{i,j}^{k}}; \tag{7}$$

$$E_{\rm te}=\sum_{i=1}^{n}\sum_{j,k=1}^{m}E_{{\rm te}_{jk}^{i}}; \tag{8}$$

$$E_{\rm we}=\sum_{j=1}^{m}\sum_{i=1}^{n}E_{{\rm we}_{ij}}=\sum_{j=1}^{m}\sum_{i=1}^{n}\beta_j\times t_{ij}; \tag{9}$$

$$\text{object1}=\min(C_{k_nm}); \tag{10}$$

$$\text{object2}=\min(E_{\rm total})=\min(E_{\rm pe}+E_{\rm ae}+E_{\rm te}+E_{\rm we}). \tag{11}$$

式 (1) 和 (2) 表示产品 k_1 的开始加工时间, 式 (3) 和 (4) 表示产品 k_i 的开始加工时间, 式 (5) 表示工件 k_i 的第 j 个工序完工时间, 式 (6) 表示加工能耗, 式 (7) 表示调整能耗, 式 (8) 表示运输能耗, 式 (9) 表示空载能耗, 式 (10) 和 (11) 表示目标函数.

2　GMOGSO 算法

基本萤火虫算法存在收敛速度慢、精度低等缺点, 且适合于连续问题求解, 为将其应用于离散多目标邻域, 需对其进行改进. 为了将算法拓展到离散域, 本文重新定义个体间距离和移动方式, 并引入交换子和交换序的概念[12].

依据交换序的概念定义萤火虫个体间方向向量 dis, 令个体 $A=[a_1,a_2,\cdots,a_n]$, 个体 $B=[b_1,b_2,\cdots,b_n]$, 则 B 到 A 的方向向量 $\text{dis}(A,B)=[d_1,d_2,\cdots,d_n]$. 其中序列 $[1\cdots i\cdots n]$ 中元素 i 与 $\text{dis}(A,B)$ 中元素 d_i 依次构成 n 对交换子 $S_i=[i,d_i]$, n 对交换子依次作用于个体 B, 使得 $A=B+S_1+\cdots+S_i+\cdots+S_n$, 一般情况下 $\text{dis}(A,B)\neq\text{dis}(B,A)$.

定义距离数乘公式为

$$c\times\text{dis}(A,B)=[d_1{}',d_2{}',\cdots,d_n{}'],\ c\in[0,1]. \tag{12}$$

其中: $\text{dis}(A,B)$ 表示从 B 到 A 的向量.

$$d_i{}'=\begin{cases} d_i,\ \text{rand}<c; \\ 0,\ \text{other}. \end{cases}$$

则 c 可以被看作从 B 向 A 移动的步长.

为了将算法应用于多目标问题, 本文重新定义个体适应度和邻居. 采用 NSGA-II 的 Pareto 排序方式[13], 将种群所有个体分级, 个体的等级可看作其适应度, 等级越高 (等级序号越低) 相应适应度越高, 种群中适应度等级比当前个体高的个体才有资格作为其邻居.

萤火虫算法收敛的本质在于个体与邻居优秀个体间的信息交流. 本文借鉴基本萤火虫算法的结构特征, 将其扩展为局部搜索过程和全局搜索过程.

2.1　局部搜索过程

GSO 中所有个体都存在感知半径, 个体邻居只能位于感知半径内, 为了增强种群多样性, 提高算法搜索能力, 改进的算法将个体感知半径扩大为整个种群. 基本萤火虫算法中移动方向选择概率只与适应度有关, 为了提高算法分散性并控制局部搜索能力, 个体移动方向选择采用如下公式:

$$P(i,j)=\frac{P'(i,j)}{\sum\limits_{j\in[1,n]}P'(i,j)},\ j\in N_1; \tag{13}$$

$$P'(i,j)=p_1\times f(i,j)+p_2\times d(i,j)+p_3\times\text{crowd}(i,j); \tag{14}$$

$$f(i,j)=1-\frac{f'(i,j)}{\max\limits_{j\in[1,n]}(f'(i,j))+1}; \tag{15}$$

$$d(i,j)=1-\frac{d'(i,j)}{\max\limits_{j\in[1,n]}(d'(i,j))+1}; \tag{16}$$

$$\text{crowd}(i,j)=1-\frac{\text{crowd}'(i,j)}{\max\limits_{j\in[1,n]}(\text{crowd}'(i,j))+1}. \tag{17}$$

其中: $f'(i,j)$、$d'(i,j)$ 和 $\text{crowd}'(i,j)$ 分别为当前个体 i 的第 j 位邻居的适应度等级、邻居与当前个体间的距离和邻居拥挤度; N_1 为个体 i 的邻居集; p_1 为适应度权重, p_1 越大算法收敛越快; p_2 为距离权重, p_2 越大算法局部搜索能力越强; p_3 为拥挤度权重, p_3 越大算

法跳出局部最优能力越强.

使用目标空间比使用决策空间有更好的效果[14], 因此邻居距离、拥挤度均在目标空间计算. 邻居距离的计算方法如下:

$$d'(i,j)=\sqrt[k]{\sum_{m=1}^{k}\left(\mathrm{fitnei}_m(i,j)-\mathrm{fit}_m(i)\right)^k}. \tag{18}$$

其中: k 为目标个数, fitnei_m、fit_m 分别为邻居解、当前解的第 m 个目标函数值. 拥挤度 $\mathrm{crowd}'(i,j)$ 采用网格法计算[15], 将目标空间划分为 N_c^k 块区域. 其中: $N_c=\sqrt[k]{\mathrm{Popsize}}$, k 为目标数, Popsize 为种群数. 每块区域内个体数即为该块内个体拥挤度 $\mathrm{crowd}'(i,j)$.

局部搜索策略保留了个体向所有优秀解移动的可能性, 导致收敛速度变慢, 如果加大选择压力, 引入精英保留策略又会使得种群多样性降低, 因此这里借鉴 ε- 支配概念[16], 引入支配松弛概念: 移动后个体 A 至少有一个目标函数值优于移动前个体 A', 则称 A' 松弛支配 A.

如果某个体是非劣解, 则没有邻居. 非劣解采用随机的移动方式, 即随机选择插入、交换、逆序操作, 插入、交换操作中移动步长 c' 表示插入操作次数, 逆序操作中 c' 表示区间长度.

局部搜索过程如下.

for $i=1:\mathrm{Popsize}$

  比较适应度等级 $\mathrm{rank}(i)$, 确定个体 $A(i)$ 和邻居集 nei;

  if $\mathrm{nei}\neq\varnothing$

    计算个体 $A(i)$ 向各个邻居 $\mathrm{nei}\{j\}$ 移动的概率 $P(i,j)$;

    用轮盘赌方式确定个体移动目标 $\mathrm{nei}\{j\}$;

    个体 $A(i)$ 以步长 c_1 向邻居 $\mathrm{nei}\{j\}$ 移动, 得到 $A'(i)=A(i)+c_1\times\mathrm{dis}(\mathrm{nei}(j),A(i))$;

  else

    个体 $A(i)$ 以步长 c' 随机移动, 得到 $A'(i)$;

  end

  if $A'(j)$ 松弛支配 $A(j)$

    $A(j)=A'(j)$;

  else

    $A(j)=A(j)$;

  end

end

2.2 全局搜索过程

局部搜索用于增强多样性, 扩大搜索空间, 但收敛较慢, 为了加快收敛速度, 在算法中引入全局搜索. 全局搜索借鉴了粒子群算法, 将个体移动目标限定为全局非劣解, 即个体的邻居只能为非劣解. 移动目标选择公式如下:

$$P(i,j)=\begin{cases}\dfrac{d(i,j)}{\sum\limits_{j\in[1,n]}d(i,j)}+\dfrac{1}{n},\ \text{解}\ j\ \text{为极值};\\ \dfrac{d(i,j)}{\sum\limits_{j\in[1,n]}d(i,j)},\ \text{其他}.\end{cases} \tag{19}$$

其中 n 为邻居个数. 当邻居 j 的任意目标为极值时, 增大个体 i 向邻居 j 移动的概率, 有利于提高解集覆盖度.

全局搜索步骤如下.

for $i=1:\mathrm{Popsize}$

  选择外部档案中个体作为邻居集 nei;

  if $\mathrm{nei}\neq\varnothing$

    计算个体 $A(i)$ 向各个邻居 $\mathrm{nei}\{j\}$ 移动的概率 $P(i,j)$;

    用轮盘赌方式确定个体移动目标 $\mathrm{nei}\{j\}$;

    个体 $A(i)$ 以步长 c_1 向邻居 $\mathrm{nei}\{j\}$ 移动,得到 $A'(i)=A(i)+c_1\times\mathrm{dis}(\mathrm{nei}(j),A(i))$;

  else

    个体 $A(i)$ 以步长 c' 随机移动, 得到 $A'(i)$;

  end

  $A(i)=A'(i)$;

end

2.3 局部搜索与全局搜索间切换

进化算法成功的关键在于平衡全局搜索与局部搜索, 本文借鉴文献 [17] 中捕食搜索策略来协调两种搜索模式. 捕食策略中常规搜索对应于本文中全局搜索部分, 区域限制搜索对应于局部搜索部分. 本文采用改善率 ir 的大小表示猎物是否出现[18]:

$$\mathrm{ir}(g)=\frac{|\{f(a)\in d(g);\exists f(b)\in d(g-1):f(a)\succ f(b)\}|}{|d(g)|}. \tag{20}$$

其中: $d(g)$ 为第 g 代非劣解集, $|d(g)|$ 为解集 $d(g)$ 中解的个数. $\mathrm{ir}(g)>0$ 表示猎物出现, 若当前采用全局搜索策略, 则切换至局部搜索, 否则不作变动; $\mathrm{ir}(g)=0$ 表示搜索无果. 令 count 为累计搜索无果次数, N 为阈值, 当前搜索策略无果且 $\mathrm{count}>N$ 时, 以概率 $1-\dfrac{1}{\mathrm{count}}$ 切换至另一搜索策略.

2.4 变步长策略

搜索初始阶段, 个体与极值间距离较大, 较大的步长有利于提高收敛速度和跳出局部最优能力, 随着迭代的进行个体与极值间距离越来越小, 较大的步长容易跳过极值, 出现震荡现象, 影响算法收敛速度和精度. 为了解决这一矛盾, 本文提出一种动态变步长

策略

$$c' = \begin{cases} U[1, \mathrm{ub}],\ \mathrm{ir}(g) \leqslant 0.1; \\ U[1, \mathrm{ub} - (\mathrm{ub} - \mathrm{lb}) \times \mathrm{ir}(g)],\ 0.1 < \mathrm{ir}(g) \leqslant 0.7; \\ \mathrm{lb},\ \mathrm{ir}(g) \geqslant 0.7; \end{cases} \tag{21}$$

$$\mathrm{ub} = \begin{cases} \dfrac{n}{3} - \dfrac{\mathrm{cal}}{\mathrm{maxcal}} \times \dfrac{n}{6},\ n > 12; \\ 2,\ n \leqslant 12; \end{cases} \tag{22}$$

$$c = c\max - \frac{\mathrm{cal}}{\mathrm{maxcal}} \times (c\max - c\min). \tag{23}$$

其中: ub、lb 分别表示随机移动步长的上界、下界, $c\max$、$c\min$ 分别表示向邻居移动步长上界、下界, cal、maxcal 分别表示当前适应度计算次数、最大适应度计算次数, n 表示产品个数, $U[a, b]$ 表示随机取 a、b 间一整数.

2.5 GMOGSO 算法流程

GMOGSO 首先使用全局搜索, 将种群推向较优区域, 再交替使用全局搜索和局部搜索进行下一阶段的寻优, 其中开始交替使用全局和局部搜索的时机由参数 percent 控制. 开始阶段的全局搜索过程为下一步的搜索提供了良好的初始解, 有利于提高收敛精度.

算法流程如下.

Step 1: 产生初始种群 A;

Step 2: 将 A 分级, $\mathrm{rank} = 1$ 的个体存入外部档案集 M;

Step 3: 初始化各代搜索策略 choose;

Step 4: while $\mathrm{cal} < \mathrm{maxcal}$

　更新搜索步长 c, 更新迭代次数 g;

　if $\dfrac{\mathrm{cal}}{\mathrm{maxcal}} < \mathrm{percent}$

　　进行全局搜索更新种群 A;

　else

　　if $(\mathrm{choose}(g)) = 1$

　　　进行全局搜索更新种群 A;

　　else

　　　进行局部搜索更新种群 A;

　　end

　end

　计算当代改善率 $\mathrm{ir}(g)$, 更新下一代搜索策略 $\mathrm{choose}(g+1)$, 更新搜索步长 c';

　对种群 A 中精英解分别采用插入、交换、逆序方式进行变邻域搜索得到种群 A';

　用 A' 中精英解取代种群 A 中最差解;

　解集 A 与外部档案 M 合并, 求出非劣解集并除去重复个体得到新的外部档案 M;

end

3 仿真实验

为了验证算法的有效性, 本文对算法进行了测试, 并与已有文献中的方法进行比较. 所有实验均在 Intel core i5 2.53 GHz、内存为 6 G 的 PC 机上进行实验, 并采用 Matlab R2014a 编程.

3.1 实验设置

本文测试用例是在 Taillard Benchmark 算例[19]的基础上加入单位时间加工能耗、单位时间空载能耗、切换能耗以及运输能耗得来的. 其中: 单位时间内空载能耗在区间 $[0.1, 0.5]$ 内随机产生, 加工能耗在区间 $[0.5, 1]$ 内随机产生, 切换能耗以及运输能耗在区间 $[0.1 \times T, 0.2 \times T]$ 内随机产生, T 表示工件所有工序的平均加工时间.

算法中, 参数通过相关文献研究并结合自身经验确定. 通过大量实验, 设置参数如下: 适应度权重 $p_1 = 2$, 距离权重 $p_2 = 1$, 拥挤度权重 $p_3 = 1.5$, 种群数量 $\mathrm{Popsize} = 200$, 无改进次数阈值 $N = 3$, 移动步长 $c\max = 0.8$, $c\min = 0.4$. 算法停止准则为适应度计算次数 $\mathrm{maxcal} = 100\,000$.

为了全面、客观地评价算法性能, 本文分别从收敛性、分散性和覆盖率去分析比较算法优劣. 收敛性采用常用的代平均距离表示[20], 分散性采用均匀分布表示[21], 覆盖率采用散布范围表示[21]. 由于难以得到算例实际 Pareto 最优前端, 本文采用以下方法获得近似前端: 所有算法分别运行 20 次, 并记录非劣解集, 从所有非劣解集中获得新的非劣解作为近似前端. 由于采用近似前端, 可能出现如图 1 所示的非劣解覆盖范围小于普通解集的情况, 导致覆盖率 $\mathrm{MS} > 1$. 本文所有性能指标均为运行 20 次的平均值.

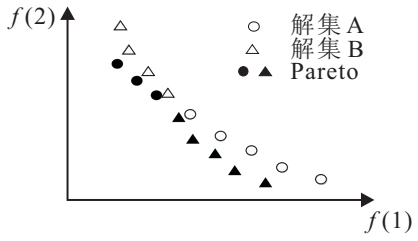

图 1　覆盖率大于 1 情况

3.2 缓冲区大小的影响

为了分析缓冲区大小对完成时间以及总能耗的影响, 分别对 $B = 0, 1, 2, 4$ 和无穷的情况进行仿真, 并采用非劣解中各个目标的平均值作为相应目标评价指标. 测试结果如表 1 所示, $B(i)$ 表示缓冲区大小为 i.

表 1 不同缓冲区大小对性能指标的影响

problem	时间					能耗				
	B(0)	B(1)	B(2)	B(4)	B(inf)	B(0)	B(1)	B(2)	B(4)	B(inf)
Car08	7633	7496	7468	7526	**7458**	10006	9380	9360	9299	**9252**
Rec01	1376	1081	1016	994	**960**	1779	1607	1544	1530	**1526**
Rec03	955	790	725	702	**701**	1523	1424	1400	1383	**1380**
Rec07	1720	1582	1594	1572	**1566**	3707	3170	3064	3077	**3055**
Rec11	1345	1237	1234	1237	**1211**	3570	2979	2851	**2807**	2822
Rec13	2035	1966	1960	**1946**	1958	6497	5458	5336	5331	**5322**
Rec29	2561	2200	2215	2106	**2040**	9565	7852	7344	7220	**7114**
Rec31	3547	3146	3034	3029	**2915**	10236	8388	7909	7666	**7615**
Rec39	6557	5669	5533	5471	**5452**	34160	27817	26131	25405	**25397**
Rec41	6557	5644	5428	5413	**5372**	35975	28338	26393	25845	**25725**

由表 1 可知, 总体上完成时间和能耗都随着缓冲区的增大越来越小, 但随着缓冲区的增大变化率越来越小, 当缓冲区由 0 增加为 1 时, 改进率最大, 这与文献 [22] 的结论一致.

缓冲区大小对完工时间以及能耗的影响一致. 产品各工序加工时间、加工工艺是确定的, 所以加工能耗和运输能耗固定, 而调整能耗与加工序列有关, 因此总能耗中只有空载能耗和完工时间相关, 缓冲区大小是通过影响空载能耗最终影响总能耗的.

3.3 切换策略有效性分析

本文算法的关键在于全局搜索与局部搜索间的切换, 单一的搜索模式不能结合其他模式的特性形成优势互补, 不恰当的切换方式也不能有效利用不同模式的优势. 例如, 采用随机切换方式时, 搜索模式切换时机不具有针对性, 搜索结果会受影响. 分别采用捕食切换策略、随机切换策略和单一全局搜索模式对不同规模算例进行仿真, 以说明切换策略的有效性 (捕食策略、随机策略和单一搜索模式分别用 A、B 和 C 表示, 缓冲区大小设为 1), 测试结果如表 2 所示.

表 2 不同策略的算法性能对比

problem	收敛性			分散性			覆盖率		
	GD(A)	GD(B)	GD(C)	UD(A)	UD(B)	UD(C)	MS(A)	MS(B)	MS(C)
Car02	62.73	53.425	**53.247**	**0.499**	0.525	0.508	**0.847**	0.779	0.726
Car05	60.272	**39.549**	47.522	0.548	0.545	**0.542**	**1.079**	0.987	0.99
Rec03	**6.549**	9.391	10.729	0.305	0.334	**0.302**	1.003	1.069	**1.13**
Rec09	**17.66**	20.56	21.246	**0.52**	0.572	0.57	**0.875**	0.743	0.732
Rec13	**28.498**	31.629	31.763	**0.498**	0.57	0.531	1.109	**1.239**	1.029
Rec19	**27.824**	41.553	30.669	**0.534**	0.584	0.606	**0.871**	0.736	0.695
Rec25	27.463	**26.079**	37.376	0.513	**0.511**	0.543	**0.81**	0.551	0.755
Rec33	**54.288**	61.315	61.579	0.701	0.609	**0.591**	0.749	0.939	**1.039**
Rec39	55.95	**43.447**	52.666	**0.484**	0.572	0.498	0.523	0.44	**0.531**

由表 2 可知, 相比于其他两种策略捕食, 搜索策略在收敛性上有一定优势. 对比随机切换策略和无切换策略 (单一搜索模式) 可知, 随机切换策略收敛性优于无切换策略, 分散性和覆盖率无明显差距. 主要原因是: 单一全局搜索模式下, 所有个体向 Pareto 前沿方向靠近, 而忽略了其他方向, 导致搜索方向单一, 使得搜索空间较小, 进而影响了搜索效果; 当搜索过程在全局搜索与局部搜索间不断切换时, 种群个体既有向最优解移动的机会也有向所有优秀个体移动的机会, 搜索过程中种群多样性得以保证, 因此收敛性也较优.

对比捕食切换模式和随机切换模式, 捕食切换模式在收敛性上有所进步. 在捕食策略中, 全局搜索只用于寻找优秀区域, 一旦进入较优区域便转入局部搜索, 相比于全局搜索, 局部搜索的搜索区域虽小但更为细致, 找到极值的概率更大; 在随机切换情况下有可能在较差区域进行局部搜索, 在较好区域进行全局搜索, 降低了搜索效率和收敛精度. 因此, 在不同的区域采用合适的搜索策略有利于提高算法性能.

3.4 与其他算法的性能比较

为了验证算法性能, 本文与其他求解多目标流水车间调度的算法进行了比较, 这些算法是 MOILS[23]、NSGA-II[13]、PGA-ALS[24]. 实验结果如表 3 所示 (缓

冲区大小设为1).

表 3　不同算法的性能指标

算例	算法	收敛性GD	分散性UD	覆盖率MS	运算时间T
Car01	GMOGSO	**36.34**	**0.37**	**1.12**	81.48
	MOILS	261.29	0.45	0.62	16.27
	NSGA-II	327.67	0.57	0.52	226.04
	PGA-ALS	467.2	0.74	0.4	193.07
Car05	GMOGSO	**34.87**	**0.53**	0.87	100.34
	MOILS	464.19	0.55	**0.88**	18.82
	NSGA-II	543.86	0.63	0.82	213.05
	PGA-ALS	886.94	0.6	0.72	226.39
Car08	GMOGSO	**23.49**	**0.71**	**0.86**	79.3
	MOILS	588.63	1	0.44	23.83
	NSGA-II	593.54	1	0.44	243.88
	PGA-ALS	674.77	0.72	0.56	220.07
Rec03	GMOGSO	**5.74**	**0.32**	**0.78**	92.18
	MOILS	11.81	0.69	0.06	24.13
	NSGA-II	29.43	0.73	0.09	221.29
	PGA-ALS	54	0.85	0.05	278.63
Rec07	GMOGSO	**11.05**	0.56	**0.91**	138.5
	MOILS	22.88	0.55	0.87	39.76
	NSGA-II	40.98	**0.54**	0.84	234.46
	PGA-ALS	77.42	0.61	0.81	226.38
Rec17	GMOGSO	**12.96**	**0.48**	**0.67**	158.11
	MOILS	36.01	0.54	0.6	52.65
	NSGA-II	43.45	0.57	0.53	244.12
	PGA-ALS	103.25	0.6	0.42	269.37
Rec23	GMOGSO	**56.89**	0.59	0.77	171.87
	MOILS	111.21	**0.58**	**0.82**	55.62
	NSGA-II	97.3	0.65	0.74	237.82
	PGA-ALS	178.04	0.61	0.67	293.45
Rec27	GMOGSO	**25.29**	**0.47**	**0.64**	196.01
	MOILS	53.55	0.54	0.53	77.06
	NSGA-II	41.99	0.56	0.48	277.84
	PGA-ALS	114.75	0.5	0.49	263.09
Rec31	GMOGSO	**48.54**	0.62	0.62	204.29
	MOILS	246.98	0.64	0.65	91.49
	NSGA-II	55.51	0.6	0.51	278.65
	PGA-ALS	100.72	**0.54**	**0.83**	305.05
Rec35	GMOGSO	**21.87**	**0.55**	**0.6**	201.95
	MOILS	336.44	0.7	0.31	92.5
	NSGA-II	32.58	0.63	0.24	279.31
	PGA-ALS	97.49	0.7	0.4	322.46
Rec39	GMOGSO	**122.34**	0.51	0.9	389.56
	MOILS	2 306.25	0.86	1.37	412.9
	NSGA-II	172.67	**0.45**	**2.19**	454.65
	PGA-ALS	347.82	0.6	1.01	436.33

由表3可知: 针对收敛性指标, GMOGSO对于所有测试问题均获得了最好的结果; 针对分散性和覆盖率指标, 本文提出的算法对于11组测试问题均有7组取得了最好结果. 因此, 总体来说, GMOGSO算法平均性能要优于文献中提出的进化算法. 单就收敛性指标而言, 对于car类问题, GMOGSO优于MOILS, MOILS优于NSGA-II, 而NSGA-II优于PGA-ALS, 但GMOGSO明显优于其他3种算法; 对于Rec类问题, NSGA-II适合求解大规模问题, MOILS适合求解小规模问题, 小规模情况下各算法间差异不大, 随着问题规模的增大, 不同算法差异越来越大. 针对运算时间进行分析, 对于以上两类问题, GMOGSO、NSGA-II和PGA-ALS均是随着问题规模增大, 运算时间T线性增长, 且GMOGSO运算时间始终小于其他两算法; MOILS的时间T随着问题规模变化大幅增加. 对MOILS单独分析, 在求解Rec类问题时, 运算时间T随着问题规模大幅增长的同时, 计算效果也越来越差, 这是由于MOILS主要采用了邻域搜索方式, 其邻域大小随着问题规模呈指数倍增长, 因此MOILS对算例的规模较为敏感.

4　结　论

本文针对带有限缓冲区的流水车间调度问题, 设计了广义多目标萤火虫算法MOGSO, 并分析了缓冲区大小分别对完成时间以及生产能耗的影响. MOGSO借鉴了NSGA-II的分级思想, 将基本萤火虫算法扩展到多目标领域, 并利用捕食搜索思想设计了全局搜索、局部搜索切换策略, 有效地提高了搜索精度. 但是, 本文未考虑缓冲区中的能耗, 虽然缓冲区能耗非主要能耗, 但对最终结果仍有一定影响, 全面考虑可能因素更有研究意义.

参考文献(References)

[1] Lin F C, Hong J S, Lin B M T. A two-machine flowshop problem with processing time-dependent buffer constraints — An application in multimedia presentations[J]. Computers & Operations Research, 2009, 36(4): 1158-1175.

[2] 于艳辉, 侯东亮. 具有缓冲区约束的流水车间调度问题综述[J]. 中国管理信息化, 2012(6): 61-63.
(Yu Y H, Hou D L. Review of flow shop scheduling problem with buffer constraints[J]. China Management Informationization, 2012(6): 61-63.)

[3] Smutnicki C. A two-machine permutation flow shop scheduling problem with buffers[J]. Operations Research Spektrum, 1998, 20(4): 229-235.

[4] Liu B, Wang L, Jin Y H. An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers[J]. Computers & Operations Research, 2008, 35(9): 2791-2806.

[5] 胡蓉, 钱斌. 一种求解随机有限缓冲区流水线调度的混合差分进化算法[J]. 自动化学报, 2009, 35(12): 1580-1586.
(Hu R, Qian B. A hybrid differential evolution algorithm for stochastic flow shop scheduling with limited buffers[J]. Acta Automatica Sinica, 2009, 35(12): 1580-1586.)

[6] Qian B, Wang L, Huang D, et al. An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers[J]. Computers & Operations Research, 2009, 36(1): 209-233.

[7] Krishnanand K N, Ghose D. A glowworm swarm optimization based multi-robot system for signal source localization[M]. Design and Control of Intelligent Robotic Systems. Springer, 2009: 49-68.

[8] Cao S, Wang J, Gu X. A wireless sensor network location algorithm based on firefly algorithm[C]. Proc of Asia Simulation Conf. Shanghai: Springer, 2012: 18-26.

[9] Krishnanand K N, Ghose D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions[J]. Swarm Intelligence, 2009, 3(2): 87-124.

[10] Zhou Y, Zhou G, Wang Y, et al. A glowworm swarm optimization algorithm based tribes[J]. Applied Mathematics & Information Sciences, 2013, 7(2): 537-541.

[11] 曾冰, 李明富, 张翼, 等. 基于萤火虫算法的装配序列规划研究[J]. 机械工程学报, 2013, 49(11): 177-184.
(Zeng B, Li M F, Zhang Y, et al. Research on assembly sequence planning based on firefly algorithm[J]. J of Mechanical Engineering, 2013, 49(11): 177-184.)

[12] Clerc M. Discrete particle swarm optimization, illustrated by the traveling salesman problem[M]. New Optimization Techniques in Engineering. Springer, 2004: 219-239.

[13] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II[J]. IEEE Trans on Evolutionary Computation, 2002, 6(2): 182-197.

[14] Talbi E G, Rahoual M, Mabed M H, et al. A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop[C]. Evolutionary Multi-Criterion Optimization. Zurich, 2001: 416-428.

[15] Coello C A C, Pulido G T, Lechuga M S. Handling multiple objectives with particle swarm optimization[J]. IEEE Trans on Evolutionary Computation, 2004, 8(3): 256-279.

[16] Laumanns M, Thiele L, Deb K, et al. Combining convergence and diversity in evolutionary multiobjective optimization[J]. Evolutionary Computation, 2002, 10(3): 263-282.

[17] Linhares A. Preying on optima: A predatory search strategy for combinatorial problems[C]. IEEE Int Conf on Systems, Man, and Cybernetics. San Diego: IEEE, 1998: 2974-2978.

[18] 郭秀萍. 多目标进化算法及其在制造系统中的应用研究[D]. 上海: 上海交通大学电子信息与电气工程学院, 2007: 25-28.
(Guo X P. MOEA and its application in manufacturing system[D]. Shanghai: School of Electronic Information and Electrical Engineering, Shanghai Jiaotong University, 2007: 25-28.)

[19] Taillard E. Benchmarks for basic scheduling problems[J]. European J of Operational Research, 1993, 64(2): 278-285.

[20] Van Veldhuizen D, Lamont G B. On measuring multiobjective evolutionary algorithm performance[C]. Proc of the 2000 Congress on Evolutionary Computation. La Jolla: IEEE, 2000: 204-211.

[21] Tan K C, Lee T H, Khor E F. Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization[J]. IEEE Trans on Evolutionary Computation, 2001, 5(6): 565-588.

[22] Wang L, Zhang L, Zheng D Z. An effective hybrid genetic algorithm for flow shop scheduling with limited buffers[J]. Computers & Operations Research, 2006, 33(10): 2960-2971.

[23] 徐建有, 董乃群, 顾树生. 带有顺序相关调整时间的多目标流水车问调度问题[J]. 计算机集成制造系统, 2013, 19(12): 3170-3176.
(Xu J Y, Dong N Q, Gu S S. Multi-objective permutation flowshop scheduling with sequence-dependent setup time[J]. Computer Integrated Manufacturing Systems, 2013, 19(12): 3170-3176.)

[24] Pasupathy T, Rajendran C, Suresh R K. A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs[J]. The Int J of Advanced Manufacturing Technology, 2006, 27(7/8): 804-815.

(责任编辑：齐 霁)