**Modeling and Simulation** 建模与仿真, 2016, 5(2), 9-15
Published Online May 2016 in Hans. http://www.hanspub.org/journal/mos
http://dx.doi.org/10.12677/mos.2016.52002

Hans汉斯

# Discrete Fruit Fly Algorithm for Multi-Objective No-Wait Flow Shop Scheduling Problem

**Yuxia Pan[1], Baoxian Jia[2]**

[1]Department of Basic Computer Education, Sanya University, Sanya Hainan
[2]The School of Computer Science, Liaocheng University, Liaocheng Shandong
Email: panyuxia2008@163.com, 36593423@qq.com

Received: Apr. 20th, 2016; accepted: May 14th, 2016; published: May 17th, 2016



## Abstract

**This paper presents a Fruit Fly Optimization Algorithm (FOA) for solving the multi-objective no-wait flow shop scheduling problem (MNFSP) with makespan and idle time criteria. Firstly, unlike the traditional FOA, the proposed algorithm applies the job-permutation-based representation. Secondly, initialization method based on the Glove generator has a uniform distribution of the solutions. Finally, a simple but effective insert search algorithm is made to improve global exploration. Computational results show that the FOA presented in this paper is very effective and efficient for the MNFSP.**

## Keywords

**Fruit Fly Optimization Algorithm, No-Wait Flow Shop Scheduling Problem, Multi-Objective**

# 多目标无等待流水线调度的离散果蝇算法

**潘玉霞[1]，贾保先[2]**

[1]三亚学院，公共计算机教学部，海南 三亚
[2]聊城大学计算机学院，山东 聊城
Email: panyuxia2008@163.com, 36593423@qq.com

文章引用: 潘玉霞, 贾保先. 多目标无等待流水线调度的离散果蝇算法[J]. 建模与仿真, 2016, 5(2): 9-15.
http://dx.doi.org/10.12677/mos.2016.52002

收稿日期：2016年4月20日；录用日期：2016年5月14日；发布日期：2016年5月17日

## 摘　要

**本文提出了一种离散多目标果蝇优化算法，求解以最大完工时间和机床空闲时间最小化为目标的无等待流水线调度问题。与传统的果蝇算法不同，首先，该算法采用基于工序的编码方式，其次，利用GLOVE发生器进行初始化，提高初始解的分散度；最后，利用简单但有效的插入方法在邻域内进化精细搜索，增强算法的全局开发能力。仿真试验表明了所提果蝇算法的有效性和高效性。**

## 关键词

**果蝇优化算法，无等待流水线调度问题，多目标**

## 1. 引言

多目标优化问题是需要同时处理多个相互冲突和相互影响的目标，一个子目标的改善有可能会引起另一个或者另几个子目标的性能降低，需要在他们中间进行协调处理。起初，多目标优化问题往往通过加权等方式转化为单目标问题，但此方法效率较低且对权值和次序较为敏感。因此，后来发展了基于Pareto最优解集(Pareto-optimal set)或非支配解集(Nondominated Set)的群体智能算法解决多目标问题。文[1]根据无等待多目标优化问题和粒子群算法的特征，采用近似 Pareto 前端的分布熵及其变化来估计种群进化状态，并依据这些进化过程反馈信息设计具有动态平衡开发能力的进化策略。文[2]为了解决无等待柔性车间调度的多目标优化问题，结合灰色失联分析和熵理论，提出灰互信息适应度分配策略，以评价 Pareto 解的优劣。

果蝇优化算法(Fruit Fly Optimization Algorithm, FOA)是由台湾博士潘文超于 2011 年提出的一类全局进化优化算法。该算法源于对果蝇觅食行为的模拟，已在自动化仓库拣选作业调度问题[3]，边坡稳定预测问题[4]，船舶操纵响应模型的辨识问题[5]等方面得到成功的应用。进化算法通过在代与代之间维持由潜在解组成的种群来实现全局搜索，这种从种群到种群的方法对于搜索多目标优化问题的 pareto 最优集是很有用的[6]。

## 2. 多目标无等待流水线调度问题 (Multi-Objective No-Wait Flow Shop Scheduling Problem, MNFSP)

### 2.1. 无等待流水线问题描述

无等待流水线调度问题要求加工的任务从开始到结束必须连续进行，即同一工件的两个相邻工序之间没有间隔。此类调度广泛存在于食品加工、炼钢、制药等生产领域。该问题可以描述为：有 $n$ 个工件 $J=\{1,2,\cdots,n\}$ 在 $m$ 台机床 $K=\{1,2,\cdots,m\}$ 上加工 $\left(O_{j,1},O_{j,2},,\cdots,O_{j,m}\right)$，工件 $j\in J$ 在 $m$ 台机床上的加工操作为。同时需要满足如下条件：所有工件启动和运输时间包含在工件的加工时间内，均可在零时刻进行加工，每个工件在各机床上的加工顺序相同。为了满足无等待的条件，推迟工件在第一台机床上的加工时间，使得工件在每台机床上的完成时间必须等于其在下一台机床上的开始加工时间，即 $O_{j,k}$ 的完成时间必须等于 $O_{j,k+1}\ (k=1,2,\cdots,m-1)$ 的开始时间。目的是得到一个可行调度 $\pi=\{\pi_1,\pi_2,\cdots,\pi_n\}$，求其 Pareto 最优解集。该问题的模型如图 1 所示。

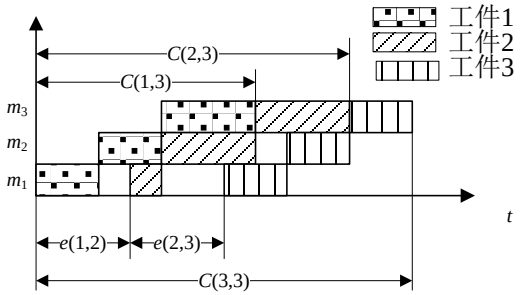


**Figure 1.** No-wait flowshop scheduling problem
**图 1.** 无等待流水线调度问题

## 2.2. 调度优化目标计算

提出以最大完工时间$\left(C_{\max}(\pi)\right)$、机床最大空闲时间$\left(Idle_{\max}(\pi)\right)$为指标对无等待调度问题进行研究。其中最大完工时间是从第一个工件开始加工计时到最后一个工件完成时间，机床空闲时间为各工件加工的空隙时间和。优化最大完工时间有利于提高生产率，降低工件的生产周期，优化机床空闲时间可以提高机床的利用率。

假设工序$\pi=\{\pi_1,\pi_2,\cdots,\pi_n\}$按机床 1 到$m$的顺序进行加工，其中$p(\pi_j,i)$为工件$\pi_j$在机床$i$上的加工时间，$e(\pi_{j-1},\pi_j)$代表工件$\pi_{j-1}$和$\pi_j$在第一台机床上的开始时间差。各工序的加工时间已知，工件$\pi_j$在机床$m$上的完成时间$C(\pi_j,m)$的计算公式为：

$$C(\pi_1,m)=\sum_{k=1}^{m}p(\pi_1,k) \tag{1}$$

$$C(\pi_j,m)=\sum_{i=1}^{j}e(\pi_{i-1},\pi_i)+\sum_{k=1}^{m}p(\pi_j,k),\quad j=2,3,\cdots,n \tag{2}$$

其中$e(\pi_{j-1},\pi_j)$的计算公式如下：

$$e(\pi_{j-1},\pi_j)=p(\pi_{j-1},1)+e(\pi_{j-1},\pi_j)=p(\pi_{j-1},1)+\max\left[0,\max_{2\le k\le m}\left\{\sum_{h=2}^{k}p(\pi_{j-1},h)-\sum_{h=1}^{k-1}p(\pi_j,h)\right\}\right] \tag{3}$$

那么工序的最小化最大完成时间计算公式如下：

$$\text{Min}\ f_1(\pi)=C_{\max}(\pi)=C(\pi_n,m)=\sum_{j=2}^{n}e(\pi_{j-1},\pi_j)+\sum_{k=1}^{m}p(\pi_n,k) \tag{4}$$

最小化最大机器空闲时间的计算公式如下：

$$\text{Min}\ f_2(\pi)=Idle_{\max}(\pi)=\sum_{k=1}^{m}C(\pi_n,k)-\sum_{i=1}^{n}p(\pi_j,k) \tag{5}$$

两个目标值算法的时间复杂度分别为是 $O(n)$，$O(mn)$。为了降低算法复杂度，提前定义变量$\delta_i$代表第一个工件与第$i$个工件在第一台机床的开工时间差

$$\delta_i=\delta_{i-1}+e(\pi_{i-1},\pi_i),i=2,3,\cdots,n,\ \delta_1=0 \tag{6}$$

$TP(\pi_i,h)$为工件$\pi_i$在第一台机床到第$h$台机床上的加工时间和

$$TP(\pi_i,h)=\sum_{k=1}^{h}p(\pi_j,k),\quad j=1,2,\cdots,n,\ \ h=1,2,\cdots,m \tag{7}$$

$$\text{Min } f_1(\pi) = \delta_n + TP(\pi_n, m) \tag{8}$$

$$\text{Min } f_2(\pi) = \sum_{k=1}^{m} C(\pi_n, k) - \sum_{i=1}^{n} TP(\pi_i, m) \tag{9}$$

通过公式(8)，(9)所示算法的时间复杂度降低为 $O(1)$，$O(n)$。

### 2.3. Pareto 最优解集

不同于单目标优化问题需要求得一个最优解，而多目标问题的优化就是寻找 Pareto 最优解集。在有限集合 $x \in \sigma$ 上，给出两个目标的调度优化问题，Minimize $f(x) = (f_1(x), f_2(x))$，若 $x_1, x_2 \in \sigma$，假定 $x_1$ 支配 $x_2$，则 $f_i(x_1) \le f_i(x_2)$，$\forall i = 1,2$。在 $\sigma$ 中如果有一个没有被任何其他解所支配，则称它为 Pareto 最优解或非支配解。所有非支配解或 Pareto 最优解的集合成为 Pareto 最优解集。

## 3. 基于 MNFSP 的离散果蝇算法

果蝇优化算法的基本原理是初始化种群的中心位置，利用敏锐的嗅觉进行搜索，即根据中心位置随机产生多个邻域解。计算各可行解的味道浓度，即评价值，然后利用视觉从中选择较好的解，更新替换中心位置，然后进行迭代寻优，以更好的进靠近食物源。FOA 在整个迭代寻优过程中，在连续空间中产生新个体的方法不适合解决 MNFSP。故运用 FOA 算法的主体流程，对其优化过程进行离散化是算法成功应用于 MNFSP 的关键。基于以上分析，提出了离散果蝇算法(Discrete Fruit Fly Optimization Algorithm DFOA)。

### 3.1. 算法编码和初始化

种群中每个果蝇个体对应问题中的一个解，即工件的完整加工序列 $\pi = \{\pi_1, \pi_2, \cdots, \pi_n\}$，如 $\pi = \{2,3,1,4\}$ 为 4 个工件的调度实例解。

果蝇初始个体产生采用 GLOVE [7]发生器产生一组分布均匀的工件排列。对于一个给定排序 $\pi$，GLOVE 发生器可得到 $n/2$ 个分散均匀的排列。随机变化给定排列，即可得到足够多的初始果蝇个体，即可构成一个分散性较好的初始种群。

### 3.2. 嗅觉和视觉搜索

影响 FOA 算法性能的核心是如何生成嗅觉阶段的邻域个体[8]。一般全局 Pareto 最优解集中在局部 Pareto 最优解附近；因此利用 Pareto 最优解集中的解产生新解的质量较高。

对于 MNFSP 来讲，插入操作是最有效的邻域解生成方法[8]。因为子代可以很好的继承父代个体的优良基因，可以充分利用非支配解信息优势，引导算法向 Pareto 最优前沿进化。嗅觉阶段在 Pareto 最优解集中随机选取一个果蝇个体，对其执行插入邻域操作步骤如下：

步骤 1：当前个体记为 $\pi$，通过对 $\pi$ 进行三次插入操作得到调度 $\pi^*$。

步骤 2：从 $\pi^*$ 中随机选取一个工件 $k$ 得到调度 $\pi'' = \{\pi''_1, \pi''_2, \pi''_{n-1}\}$ 用公式(8)(9)分别计算 $C_{\max}(\pi'')$ 和 $Idle_{\max}(\pi'')$。

将工件 $k$ 插入到所有可能的位置 $k'$，$k' \in \{1,2,\cdots,n\} \wedge k' \notin \{k-1, k\}$ 得到调度 $\pi'$，如图 2 所示。

指标计算公式如下：

$$C_{\max}(\pi') = \begin{cases} C_{\max}(\pi'') + e(\pi_k, \pi''_0) & k' = 0 \\ C_{\max}(\pi'') + e(\pi''_{k'-1}, \pi_k) + e(\pi_k, \pi''_{k'}) - e(\pi''_{k'-1}, \pi''_{k'}) & 0 < k' < n \\ C_{\max}(\pi'') + e(\pi''_{n-1}, \pi_k) + TP(\pi_k) - TP(\pi''_{n-1}) & k' = n \end{cases} \tag{10}$$

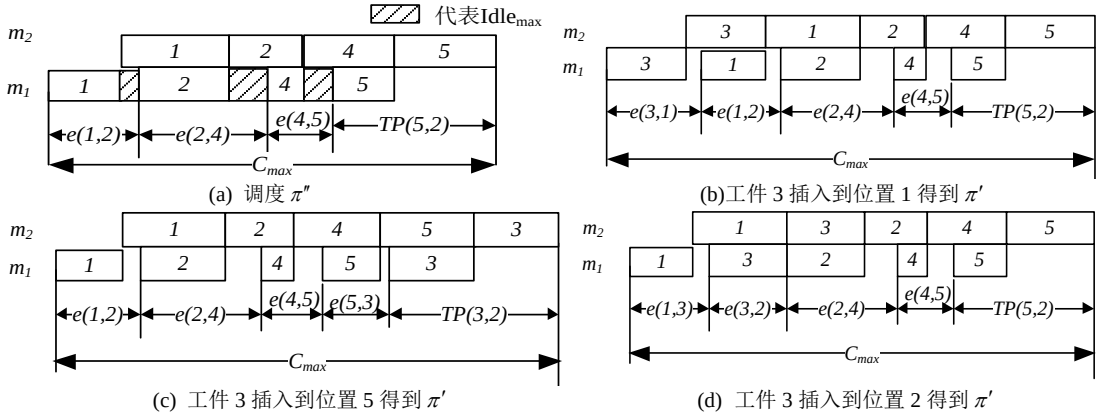


**Figure 2.** Fast calculation method for makespan and idle time of insert the neighborhood
**图 2.** 插入邻域完工时间和空闲时间快速计算方法

$$Idle_{\max}(\pi') = \begin{cases} Idle_{\max}(\pi'') + m * e(\pi_k, \pi_0'') - TP(\pi_k, m) & k' = 0 \\ Idle_{\max}(\pi'') + m * e(\pi_{k'-1}'', \pi_k) m * e(\pi_k, \pi_{k'}'') - m * e(\pi_{k'-1}'', \pi_{k'}'') & 0 < k' < n \\ -TP(\pi_k, m) Idle_{\max}(\pi'') + m * e(\pi_{n-1}'', \pi_k) - TP(\pi_k, m) & \\ \quad -\sum_{i=1}^{m} TP(\pi_{n-1}'', i) + \sum_{i=1}^{m} TP(\pi_k, i) & k' = n \end{cases} \tag{11}$$

步骤 3：$\theta$ 代表临时非支配解集，保存插入过程中得到的非支配解。

步骤 4：若 $\theta$ 中有解没有被 AS 中的解支配，则更新 AS。

步骤 5：若 $\theta$ 中有解支配 $\pi$，则替换之；若互不支配，则随机选择一个替换 $\pi$。

确定新解是否为非支配解，若为非支配解，则更新 AS，同时选择较好的解替换当前解。

## 4. DFOA 算法流程

基于上述设计，DFOA 算法的流程步骤如下所示：

步骤 1：设置参数，初始化种群。

步骤 2：评价果蝇个体，并判断是否为非支配解，更新 AS。

步骤 3：在 AS 中随机选择一个非支配解，利用算法的嗅觉和视觉部分产生新的个体，并更新 AS。

步骤 4：判断终止条件是否满足，是则转到步骤 5，否则转到步骤 3。

步骤 5：算法终止。

## 5. 仿真实验

### 5.1. 实验设置

将本文所提算法 DFOA 与文献[9] ESFLA 进行比较，通过不同规模的解决 6 个标准问题进行有效性验证。几种算法都在处理器为 Intel(R)Core i3 2.13 GHZ、内存为 2G 的 PC 机上，采用 C++编码进行测试。对于每个算例，两种算法分别独立运行 20 次，$S_j\ (j=1,2)$ 为第 $j$ 个算法得到的非支配解集，从中选出非支配解集作为参考非支配解集 $S^* = \cup S_j$。算法最大运行时间为 $T = 10 \times m \times n$ 微秒。采用两个性能指标来评价算法所得到的非支配解集：距 Pareto 边界的平均距离 $Dl_R$、非支配解的比率 $R_{NDS}S_j$。

### 5.2. 仿真结果对比

对文献[9]提出的 ESFLA 算法进行修改，按文中规定的参数进行设置，并将其应用于求解基于最大

**Table 1.** The comparation of $Dl_R$ between DFOA and ESFLA
**表 1.** DFOA、ESFLA 算法的 $Dl_R$ 比较

| 算例 | | DFOA | | | ESFLA | | |
|---|---|---|---|---|---|---|---|
| 名称 | $n \times m$ | AVG | MIN | MAX | AVG | MIN | MAX |
| Car01 | 11 × 5 | 0.15 | 0.00 | 2.94 | 1.51 | 2.24 | 5.14 |
| Car07 | 7 × 7 | 0.04 | 0.03 | 0.79 | 0.05 | 0.00 | 1.03 |
| Hel1 | 100 × 10 | 0.10 | 0.00 | 1.96 | 0.79 | 0.66 | 2.07 |
| Hel2 | 20 × 10 | 0.07 | 0.00 | 1.47 | 0.99 | 0.49 | 3.14 |
| Rec35 | 50 × 10 | 0.02 | 0.00 | 0.43 | 0.35 | 0.17 | 3.42 |
| Rec37 | 75 × 20 | 0.04 | 0.00 | 0.72 | 0.12 | 0.03 | 2.35 |
| Mean | | 0.07 | 0.01 | 1.39 | 0.64 | 0.60 | 2.86 |

**Table 2.** The comparation of $R_{NDS}S_j$ between DFOA and ESFLA
**表 2.** DFOA 和 ESFLA 算法的 $R_{NDS}S_j$ 比较

| 算例 | | DFOA | | | ESFLA | | |
|---|---|---|---|---|---|---|---|
| 名称 | $n \times m$ | AVG | MIN | MAX | AVG | MIN | MAX |
| Car01 | 11 × 5 | 0.99 | 0.97 | 1.00 | 0.68 | 0.56 | 0.94 |
| Car07 | 7 × 7 | 0.96 | 0.92 | 1.00 | 0.89 | 0.68 | 1.00 |
| Hel1 | 100 × 10 | 0.85 | 0.59 | 0.98 | 0.78 | 0.32 | 0.89 |
| Hel2 | 20 × 10 | 0.78 | 0.51 | 1.00 | 0.66 | 0.54 | 0.78 |
| Rec35 | 50 × 10 | 0.71 | 0.19 | 0.92 | 0.65 | 0.40 | 0.82 |
| Rec37 | 75 × 20 | 0.79 | 0.13 | 0.98 | 0.78 | 0.31 | 0.89 |
| Mean | | 0.85 | 0.58 | 0.97 | 0.76 | 0.52 | 0.87 |

完工时间和最大机床空闲时间的多目标无等待流水线调度问题。两种算法采用相同的终止条件。AVG、MIN 和 MAX 分别代表 20 次仿真实验目标值的平均值、最小值和最大值。结果如表 1，表 2 所示。

由于真正的非支配解很难搜索到，有必要对所得解集计算 $Dl_R$，距离越近，得到的解质量就越高。由表 1 可知，DFOA 算法的平均 $Dl_R$，最小 $Dl_R$，以及最大 $Dl_R$ 的值分别是 0.07，0.01，1.39，均小于 ESFLA 算法的 0.64，0.60，2.86。因此，相对于平均距离而言，DFOA 算法性能优于 ESFLA。

算法 DFOA 优于 ESFLA 的另一个方面表现在非支配解的比率上。由表 2 可以看出，DFOA 算法中平均有 85%没有被 $S^*$ 中的其他解所支配，而 ESFLA 算法中只有 76%。

## 6. 结论

综上所述，DFOA 算法之所以优于 ESFLA，因为 DFOA 算法在算法中执行过程中保留了所以的非支配解，并充分利用了非支配解的优势，在其基础上用插入方法产生邻域解，更好的指导了算法的进化方向，增强算法全局搜索的能力。

## 基金项目

海南省教育厅科研项目(Hnky2015-51, Hnky2015-55)；三亚市院地科技合作项目(2015YD57, 2015YD11)。

## 参考文献 (References)

[1] 胡旺, Gary YEN, 张鑫. 基于 Pareto 熵的多目标粒子群优化算法[J]. 软件学报, 2014, 24(5): 1025-1050.

[2] 毕孝儒, 张黎黎, 贺拴, 等. 面向无等待多目标柔性车间调度问题的遗传蜂群优化算法[J]. 研究与开发, 2015(8): 11-16.

[3] 刘志雄, 王雅芬, 张煜. 多种群果蝇优化算法求解自动化仓库拣选作业调度问题[J]. 武汉理工大学学报, 2014, 36(3): 71-77.

[4] 王海军, 涂凯, 闫晓荣. 基于果蝇优化算法的 GRNN 模型在边坡稳定预测中的应用[J]. 水电能源科学, 2015, 33(1): 124-126.

[5] 王雪刚, 邹早建. 基于果蝇优化算法的船舶操纵响应模型的辨识[J]. 大连海事大学学报, 2012, 38(3): 1-4.

[6] 公茂果, 焦李成, 杨咚咚, 等. 进化多目标优化算法研究[J]. 软件学报, 2009, 20(2): 271-289.

[7] Glover, F. (1998) A Template for Scatter Search and Path Reclinking. Artificial Evolution. *Lecture Notes in Computer Science*, **1363**, 1-51.

[8] 郑晓龙, 王凌, 王圣尧. 求解置换流水线调度问题的混合离散果蝇算法[J]. 控制理论与应用, 2014, 31(2): 159-164.

[9] 潘玉霞, 潘全科, 李俊青. 蛙跳优化算法求解多目标无等待流水线调度[J]. 控制理论与应用, 2011, 28(10): 1363-1370.