# 增强人工蜂群算法求解半导体最终测试调度问题

吕 阳[1,2],钱 斌[1,2],胡 蓉[1,2],张梓琪[1]
(1.昆明理工大学信息工程与自动化学院,云南昆明 650500;2.昆明理工大学云南省人工智能重点实验室,云南昆明 650500)

**摘 要:** 本文提出一种增强人工蜂群算法(Enhanced Artificial Bee Colony,EABC),用于最小化半导体最终测试调度问题(Semiconductor Final Testing Scheduling Problem,SFTSP)的最大完工时间.该算法采用混合启发式方法初始化种群,并利用前插式解码策略来提高初始解的质量.在算法搜索阶段设计多种基于问题性质的探索策略和基于贝叶斯网络的概率模型对问题解空间进行深度与宽度的协同搜索.此外,提出基于重启策略的种群更新机制以加强算法跳出局部最优的能力.实验部分构造多种对比算法来验证EABC中各关键环节的有效性.通过基于实例的数值仿真以及与NFOA(Novel Fruit fly Optimization Algorithm)、KMEA(Knowledge-based Multi-agent Evolutionary Algorithm)和CCIWO(Cooperative Co-evolutionary Invasive Weed Optimization)的算法比较验证了EABC的有效性和鲁棒性.

**关键词:** 半导体最终测试;人工蜂群算法;启发式规则;贝叶斯网络;多策略融合;概率模型;排序模型
**中图分类号:** TP391.9;TN406 **文献标识码:** A **文章编号:** 0372-2112(2021)09-1708-08
**电子学报URL:** http://www.ejournal.org.cn **DOI:** 10.12263/DZXB.20210039

# Enhanced Artificial Bee Colony Algorithm to Solve Semiconductor Final Test Scheduling Problem

LÜ Yang[1,2],QIAN Bin[1,2],HU Rong[1,2],ZHANG Zi-qi[1]
(1.*Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, Yunnan* 650500, *China*;
2.*Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming, Yunnan* 650500, *China*)

**Abstract:** This paper proposed an enhanced artificial bee colony algorithm (EABC) to minimize the maximum completion time of the semiconductor final test scheduling problem (SFTSP). EABC used hybrid heuristic methods to initialize the population, and used forward interpolation decoding strategies to improve the quality of the initial solution. A variety of exploration strategies based on the problem features and a Bayesian network-based probability model were designed to conduct a depth and width search of the problem solution space. In addition, a population update mechanism based on the restart strategy was proposed to strengthen the algorithm´s ability to jump out of the local optimum. In the experimental part, multiple comparison algorithms were constructed to verify the effectiveness of the EABC's key components. Simulation results based on some instances and comparisons with NFOA(Novel Fruit fly Optimization Algorithm)、KMEA(Knowledge-based Multi-agent Evolutionary Algorithm) and CCIWO(Cooperative Co-evolutionary Invasive Weed Optimization) demonstrated the effectiveness and robustness of the proposed algorithm.

**Key words:** semiconductor final test;artificial bee colony algorithm;heuristic rules;Bayesian network;multi-strategy integration;probability model;permutation-based model

## 1 引言

芯片制造业是我国最重要的高新技术产业之一,其发展水平决定了我国信息化、数字化的速度和高度[1].半导体最终测试调度问题(Semiconductor Final Testing Scheduling Problem,SFTSP)作为半导体产品的最终测试环节,通过对封装好的半导体产品进行性能测试和筛选,剔除不合格产品,来保证产品的优质率.自20世纪90年代初以来,SFTSP受到众多学者的广泛关注和研究.Uzsoy等[2,3]和Wu等[4]设计了多条启发式规则求解SFTSP.随后,Wu等[5]和Hao等[6]采用元启发

收稿日期:2020-12-29;修回日期:2021-02-10;责任编辑:覃怀银
基金项目:国家自然科学基金(No. 51665025, No. 61963022)

算法对SFTSP进行求解.近年来,Zheng[7]和Sang[8]设计了改进的元启发式算法对SFTSP进行求解.并取得了不错的效果.

人工蜂群算法(Artificial Bee Colony algorithm,ABC)作为一种经典的元启发算法,在函数优化、生产调度等领域得到了越来越多的研究和应用.Lu等[9]采用ABC算法求解不相关并行机调度问题.Li等[10]设计混合ABC算法求解分布式流水线问题.Gao等[11]和Gong等[12]采用ABC算法求解柔性作业车间调度问题(FJSP),并取得了不错的效果,本文所研究的SFTSP本质上可以建模为一类带有多资源约束的FJSP.因此,采用ABC算法求解SFTSP是有意义的.目前,ABC的改进工作主要集中在算法各阶段的搜索机制上.Li等[13]通过加入邻域搜索算子改进了ABC算法的雇佣蜂阶段.Banharnsakun等[14]设计优质解共享策略,以改进ABC算法的观察蜂阶段.Li等[15]提出了一种GABC算法,改进了ABC算法的侦察蜂阶段.文献调研可知,现有ABC相关改进算法大多注重于在算法的各个阶段构造邻域算子来对种群中所有个体或者部分优质个体执行局部搜索操作,从而提升算法自主搜索能力,较少考虑种群优势信息的积累和算法跳出局部最优能力的加强.

本文针对SFTSP,建立以最小化最大完工时间为优化目标的排序模型,并提出一种增强人工蜂群算法进行求解.在EABC雇佣蜂阶段设计多种搜索策略来保证算法搜索的深度;观察蜂阶段引入基于贝叶斯网络的概率模型学习并积累种群优势信息;侦察蜂阶段设计基于重启策略的种群更新机制来加强算法跳出局部最优的能力.

## 2　问题介绍

### 2.1　问题描述及相关符号定义

SFTSP描述如下:$n$个测试工件(半导体元件)需要在$m$台机器上进行性能测试,每个工件$J_i$必须通过一组操作$\{O_{i1}, O_{i2}, \cdots, O_{in_i}\}$,每个操作$O_{ij}$允许在多台可用机器$M_i \in M$上执行测试.假设操作$O_{ij}$选择机器$M_i$进行测试,机器$M_i$需要一台测试机器、一台处理机器和一台附属机器进行配合.同一组的测试资源都是相同的,不同组的测试资源不同.对于机器$M_i$,所需要的资源是预先确定的,在测试过程中不能互换使用,只有当所需的测试机器、处理机器和附属机器都可用,并且作业到达机器时,测试过程才能开始.此外,还需要满足以下约束:(1)测试过程不允许被中断,在整个测试过程中,需要对机器和资源进行独占使用;(2)一个作业在任何时候最多只能在一台机器上测试;(3)机器要测试一个工件,需要预先设置一个与测试过程分离的时间;(4)所有测试工件、机器和资源在初始时刻都是可用的,不同工件的操作之间不存在优先约束.工件的测试时间和测试操作已提前给出.所涉及到相关符号定义如表1所示.

**表1　符号定义表**

| 符号 | 定义 |
|---|---|
| $n_i$ | 第$i$个工件的操作数,$i = \{1, 2, \cdots, n\}$ |
| TS | 所有工件的工序总数, $\mathrm{TS} = \sum_{i=1}^{n} N_i$ |
| $M$ | 机器集合,$M = \{M_1, M_2, \cdots, M_m\}$ |
| $\boldsymbol{\pi}$ | 问题的解序列,$\boldsymbol{\pi} = (\boldsymbol{\pi}^s, \boldsymbol{\pi}^a)$ |
| $N_{T_i}$ | 第$T_i$类测试机器的总数 |
| $N_{H_i}$ | 第$H_i$类处理机器的总数 |
| $N_{A_i}$ | 第$A_i$类附属机器的总数 |
| $\mathrm{SM}_i$ | 机器$M_i$所需要的资源配置,$\mathrm{SM}_i = \{T_a, H_b, A_c\}$ |
| $\boldsymbol{\pi}^i$ | 为第$i$台设备上所测试工件基于工序的排列,$\boldsymbol{\pi}^i = (\pi_1^i, \pi_2^i, \cdots, \pi_{Q_i}^i)$ |
| $P(\pi_k^i)$ | 工序$\pi_k^i$所需的测试时间,$k = 0, 1, \cdots, Q_i$;$P(\pi_0^i) = 0$ |
| $S(\pi_k^i)$ | 工序$\pi_k^i$的开始时间,$k = 0, 1, \cdots, Q_i$;$S(\pi_0^i) = 0$ |
| $C(M_i)$ | 某一时刻机器$M_i$上最后一个工件的完工时间 |
| $C(\pi_k^i)$ | 工序$\pi_k^i$的完工时间 |
| $\mathrm{pm}(\pi_k^i)$ | 工序$\pi_k^i$前一次测试所用设备号 |
| $\mathrm{pk}(\pi_k^i)$ | 工序$\pi_k^i$在前一次测试设备上的排序中的位置 |
| St | 相关设置时间,表示同一个工件的上一个工序从机器$j$转移到机器$i$所需要的设置时间 |
| $R_k^i$ | 机器$i$在加工工序$\pi_k^i$时所需要的资源准备完毕时刻 |
| $\mathrm{Pf}_k^i$ | $\pi_{k-1}^i$与$\pi_k^i$之间的机器空闲时间 |
| $\mathrm{Sf}_k^i$ | 工序$\pi_k^i$进行插入操作的开始加工时间 |

### 2.2　SFTSP排序模型

SFTSP的优化目标是找到一个最优工件操作序$\boldsymbol{\pi}^{s*}$,使其按机器分配序$\boldsymbol{\pi}^{a*}$进行测试时,完工时间$C_{\max}$最小,即

$$C_{\max} = \max\{C(\pi_{Q_1}^1), C(\pi_{Q_2}^2), \cdots, C(\pi_{Q_m}^m)\} \tag{1}$$

SFTSP排序模型如下:

如果$\mathrm{pk}(\pi_k^i)$=0,则

$$C(\pi_k^i) = \begin{cases} \max\{S(\pi_k^i), R_k^i\} + P(\pi_k^i), & k = 1 \\ \max\{C(\pi_{k-1}^i) + \mathrm{St}(\pi_{\mathrm{pk}(\pi_k^i)}^{\mathrm{pm}(\pi_k^i)}, \pi_k^i), R_k^i\}, & k \neq 1 \end{cases} \tag{2}$$

否则

$$C(\pi_k^i) = \begin{cases} \max\{C(\pi_{\mathrm{pk}(\pi_k^i)}^{\mathrm{pm}(\pi_k^i)}), R_k^i\} + P(\pi_k^i), & k = 1 \\ \max\{C(\pi_{k-1}^i) + \mathrm{St}(\pi_{\mathrm{pk}(\pi_k^i)}^{\mathrm{pm}(\pi_k^i)}, \pi_k^i), \\ \qquad C(\pi_{\mathrm{pk}(\pi_k^i)}^{\mathrm{pm}(\pi_k^i)}), R_k^i\}, & k \neq 1 \end{cases} \tag{3}$$

$$C_{\max}(\boldsymbol{\pi}^*) = \min_{m \subset \Omega} C_{\max}(\boldsymbol{\pi}) \tag{4}$$

$$\boldsymbol{\pi}^* = \arg\{C_{\max}(\boldsymbol{\pi})\} \rightarrow \min, \forall \boldsymbol{\pi} \subset \boldsymbol{\Omega} \tag{5}$$

其中

$$C(\boldsymbol{\pi}_k^i) = S(\boldsymbol{\pi}_k^i) + P(\boldsymbol{\pi}_k^i) \tag{6}$$

$$\text{如果} N_{T_a} > 0, N_{H_b} > 0, N_{T_a} > 0, \text{则} R_k^i = 0 \tag{7}$$

否则,令 $\text{temp}_1, \text{temp}_2, \text{temp}_3 = 0$

如果 $N_{T_a} = 0$,则

$$\text{temp}_1 = \min\{C(M_1^{T_a}), C(M_2^{T_a}), \cdots, C(M_k^{T_a})\} \tag{8}$$

如果 $N_{H_b} = 0$,则

$$\text{temp}_2 = \min\{C(M_1^{H_b}), C(M_2^{H_b}), \cdots, C(M_k^{H_b})\} \tag{9}$$

如果 $N_{A_c} = 0$,则

$$\text{temp}_3 = \min\{C(M_1^{A_c}), C(M_2^{A_c}), \cdots, C(M_k^{A_c})\} \tag{10}$$

$$R_k^i = \max\{\text{temp1}, \text{temp2}, \text{temp3}\} \tag{11}$$

其中,式(1)至式(3)为最大完工时间的计算公式,式(4)和式(5)表示在所有解序列集合 $\boldsymbol{\Omega}$ 中找到最优解序列 $\boldsymbol{\pi}^*$,使得 $C_{\max}(\boldsymbol{\pi})$ 最小,式(6)为工序的完工时间计算公式,式(7)至式(11)为机器所需要的资源准备完毕时刻.

## 3 增强人工蜂群算法(EABC)

### 3.1 编码与解码

#### 3.1.1 SFTSP的双矢量编码

EABC中食物源的编码采用常见的双矢量编码方式[9],种群中的每个食物源向量可以表示为 $\boldsymbol{\pi} = (\boldsymbol{\pi}^s, \boldsymbol{\pi}^a)$.

#### 3.1.2 基于问题性质的前插式解码策略

基于所提排序模型以及相关定理进行解码操作.

**定理1** 设 $\boldsymbol{\pi}_a^{k1}$ 是工件 $i$ 的第一个工序,当满足:① $N_{T_{k1}} > 0, N_{H_{k1}} > 0, N_{A_{k1}} > 0$;② $\text{Pf}_1^{k1} \geqslant P(\boldsymbol{\pi}_a^{k1})$ 时,$\boldsymbol{\pi}_a^{k1}$ 可以进行前插操作,且插入后的 $\text{Sf}_1^{k1} = 0$,其中,$\text{Pf}_1^{k1} = P(\boldsymbol{\pi}_{\text{pk}(\pi_1^{k1})}^{\text{pm}(\pi_1^{k1})})$.

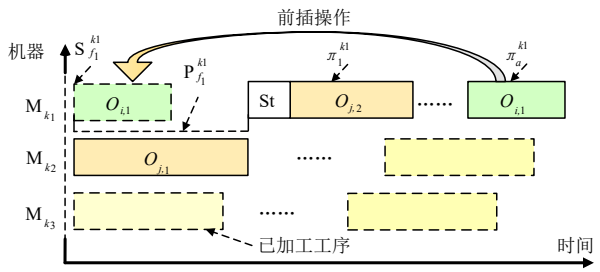


图1 工序前插情况1

**定理2** 设 $\boldsymbol{\pi}_k^{k1}$ 是工件 $i$ 的第 $a$+1 道工序,$\boldsymbol{\pi}_l^{k1}$ 是工件 $j$ 第 $a$+1 道工序,且 $C(\boldsymbol{\pi}_{\text{pk}(\pi_k^{k1})}^{\text{pm}(\pi_k^{k1})}) \geqslant C(\boldsymbol{\pi}_{l-1}^{k1})$,令 $\text{temp} = C(\boldsymbol{\pi}_{\text{pk}(\pi_k^{k1})}^{\text{pm}(\pi_k^{k1})}) - C(\boldsymbol{\pi}_{l-1}^{k1})$,当满足:

①$N_{T_{k1}} > 0, N_{H_{k1}} > 0, N_{A_{k1}} > 0$;

②$\text{Pf}_l^{k1} - \text{temp} \geqslant P(\boldsymbol{\pi}_k^{k1}) + \text{St}(\boldsymbol{\pi}_{\text{pk}(\pi_k^j)}^{\text{pm}(\pi_k^i)}, \boldsymbol{\pi}_k^i)$时,$\boldsymbol{\pi}_k^{k1}$ 可进行前插操作(插入到 $\boldsymbol{\pi}_l^{k1}$ 前),插入位置的开始时间为:

$$\text{Sf}_k^{k1} = C(\boldsymbol{\pi}_{\text{pk}(\pi_k^{k1})}^{\text{pm}(\pi_k^{k1})}) + \text{St}(\boldsymbol{\pi}_{\text{pk}(\pi_k^{k1})}^{\text{pm}(\pi_k^{k1})}, \boldsymbol{\pi}_k^{k1}),$$

其中,$\text{Pf}_l^{k1} = C(\boldsymbol{\pi}_{\text{pk}(\pi_l^{k1})}^{\text{pm}(\pi_l^{k1})}) - C(\boldsymbol{\pi}_{l-1}^{k1})$

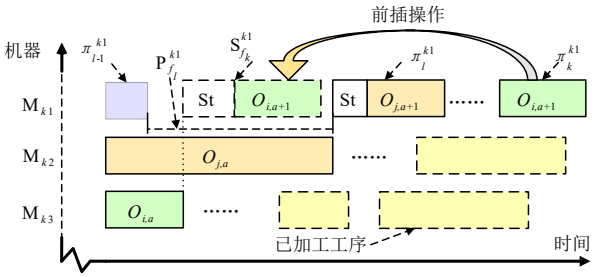


图2 工序前插情况2

**定理3** 设 $\boldsymbol{\pi}_k^{k1}$ 是工件 $i$ 的第 $a$+1 道工序,$\boldsymbol{\pi}_l^{k1}$ 是工件 $j$ 的第 $a$+1 道工序,且 $C(\boldsymbol{\pi}_{\text{pk}(\pi_k^{k1})}^{\text{pm}(\pi_k^{k1})}) < C(\boldsymbol{\pi}_{l-1}^{k1})$,当满足:① $N_{T_{k1}} > 0, N_{H_{k1}} > 0, N_{A_{k1}} > 0$;

②$\text{Pf}_l^{k1} \geqslant P(\boldsymbol{\pi}_k^{k1}) + \text{St}(\boldsymbol{\pi}_{\text{pk}(\pi_k^j)}^{\text{pm}(\pi_k^i)}, \boldsymbol{\pi}_k^i)$时,可进行前插操作插入位置的开始时间为:

$$\text{Sf}_k^{k1} = C(\boldsymbol{\pi}_{l-1}^{k1}) + \text{St}(\boldsymbol{\pi}_{\text{pk}(\pi_k^{k1})}^{\text{pm}(\pi_k^{k1})}, \boldsymbol{\pi}_k^{k1}),$$

其中,$\text{Pf}_l^{k1} = C(\boldsymbol{\pi}_l^{k1}) - C(\boldsymbol{\pi}_{l-1}^{k1})$

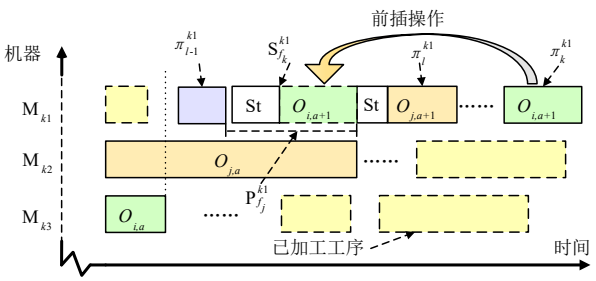


图3 工序前插情况3

### 3.2 混合启发式规则

较好的初始化解可使算法搜索集中在解空间中有潜力的区域进行,提高算法搜索性能.因此本文分别针对工序排列和机器分配设计了混合启发式规则用于生成初始解.

工件排序阶段采用文献[16]所设计的启发式规则生成 $\boldsymbol{\pi}^s$.

为了较好的平衡每台机器和每种资源的负荷情况,机器分配阶段各工序根据累积负荷因子分配负荷最小的测试机器或资源设备,并在后续过程中,更新机器的累积负荷因子LF和资源的累积负荷因子SF.详细过程如算法1.

**算法1　机器分配规则**

**输入:** $\pi^s$;

**输出:** $\pi^a$;

初始化LF和SF,令$j$=1;

repeat

　随机生成$r_m$ = rand(0, 1);

　If $r_m \in [0, 0.4)$ then

　　令$k$ = min LF($i$),其中$i$为可加工工序$\pi_j^s$的机器集合,则$\pi_j^a = k$,并更新LF和SF,$j = j + 1$;

　Else If $r_m \in [0.4, 0.8)$ then

　　令$k$ = min SF($i$),其中$i$为可加工工序$\pi_j^s$的机器集合,则$\pi_j^a = k$,并更新LF和SF,$j = j + 1$;

　Else If $r_m \in [0.8, 1)$ then

　　令$k = i$,其中$i$为可加工工序$\pi_j^s$的机器集合,则$\pi_j^a = k$,并更新LF和SF,$j = j + 1$;

Until $j$>TS;

### 3.3　多策略融合的雇佣蜂阶段

雇佣蜂阶段设计了多种搜索策略实现在每一个食物源附近的细致搜索.结合多种搜索策略,提出个体更新公式:

$$I_i^{\text{gen}+1} = \{[\text{repair}\odot(\text{swap}\odot I_i^{\text{gen}})(\pi_i^a)]^{r_m}\odot \text{rule}\}$$

其中,依次执行的三个操作分别为$\text{swap}\odot I_i^{\text{gen}}$,$\text{repair}\odot(\text{swap}\odot I_i^{\text{gen}})(\pi_i^a)$和$[\text{repair}\odot(\text{swap}\odot I_i^{\text{gen}})(\pi_i^a)]^{r_m}\odot$ rule.

#### 3.3.1　$\text{swap}\odot I_i^{\text{gen}}(\pi_i^s)$描述

$\text{swap}\odot I_i^{\text{gen}}(\pi_i^s)$表示随机选择操作序列上$\xi$个位置的序号执行随机交换操作.本文针对$\xi$设计自适应调节机制如式(12)所示:

$$\xi(\text{gen}) = \text{int}\{[\xi(0) - \xi(\text{gen}_{\max})]\times\frac{\text{gen}_{\max} - \text{gen}}{\text{gen}_{\max}} + \xi(\text{gen}_{\max})]\} \tag{12}$$

执行完此节操作后,个体更新公式变为如下所示:

$$I_i^{\text{gen}+1} = [\text{repair}\odot\overline{I_i^{\text{gen}}}(\pi_i^a)]^{r_m}\odot \text{rule} \tag{13}$$

#### 3.3.2　$\text{repair}\odot\overline{I_i^{\text{gen}}}(\pi_i^a)$描述

执行交换操作之后,同一个工件的不同工序所对应的机器分配序会发生错乱,导致不满足机器与工序之间的加工约束关系,需要对交换后的机器分配序列执行修复操作,如图4所示.执行$\text{repair}\odot\overline{I_i^{\text{gen}}}(\pi_i^a)$后,得到式(14):

$$I_i^{\text{gen}+1} = [\overline{I_i^{\text{gen}}}]^{r_m}\odot \text{rule} \tag{14}$$

#### 3.3.3　$I_i^{\text{gen}+1} = [\overline{I_i^{\text{gen}}}]^{r_m}\odot$ rule描述

为了进一步对交换后的解序列进行细致搜索,本节设计了两条分配策略用于实现解序列的更新,过程如下.

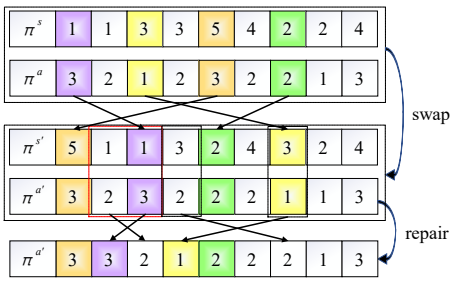


图4　交换和修复操作示意图

$$I_i^{\text{gen}+1} = [\overline{I_i^{\text{gen}}}]^{r_m}\odot \text{rule} = \begin{cases} \overline{I_i^{\text{gen}}}\odot \text{rule\_1}, r_m > 0.5 \\ \overline{I_i^{\text{gen}}}\odot \text{rule\_2}, r_m \leqslant 0.5 \end{cases} \tag{15}$$

rule_1:确定负荷最大的前$k$个机器,随机选择这些机器上的一道工序进行重新分配.

rule_2:随机生成一个1到$m$的整数$K$,选择机器$K$上的工序对其进行重新分配,若可加工的机器只有一台则不重新分配.

### 3.4　基于贝叶斯网络的观察蜂阶段

作为蜂群协同进化过程,观察蜂通过相互学习从而获取优质解信息,考虑到传统交叉操作所带来的模式破坏问题,本节采用贝叶斯网络模型学习和积累优质解信息,并通过采样概率模型生成新解来更新种群.

观察蜂阶段根据食物源的适应度值采用锦标赛选择策略从种群中选取PS × $\theta$个优质食物源,然后,基于所选择的个体,分别构建对应于工件操作序和机器分配序的贝叶斯网络,其中,贝叶斯网络的层数为TS,每层的节点数为$m_1$,贝叶斯网络由TS × $m_1$个节点所构成,节点$N_{i,j}$表示解序列中第$i$个位置存放的编号为$j$,在SFTSP中,如果是工件操作序列,则$m_1$取值为$n$,如果是机器分配序列,则$m_1$取值为$m$.从$N_{i,j}$到$N_{i+1,k}$的定向弧所具有的权重为种群中精英个体中符合该子序的数量.

针对概率模型采样生成新个体,一般选用轮盘赌策略,但对于离散工件操作序或机器分配序采样时存在如下问题:

(1)部分定向弧概率为0,则意味在更新食物源时在此位置无法得到对应的工件序号,使得算法在后续迭代过程中有可能陷入局部最优.

(2)面对离散问题进行采样时,直接轮盘赌策略往往容易造成节点重复选中而产生非法解的情况.

(3)当采样生成机器分配序列时,会出现工序在所分配的机器上无法加工的情况.

针对上述问题,本文从三个方面对贝叶斯概率模型进行了改进.针对问题1,在初始化贝叶斯网络阶段时,赋予所有可行的定向弧权重1/(PS × $\theta$).这一操作一定程度上提升了种群个体的多样性,有利于防止算法过早收敛陷入局部最优.针对问题2和问题3,结合基

于贝叶斯概率模型的特点，设计了一种改进概率模型更新方式，进而保证轮盘赌策略的高效运行．基于贝叶斯网络的观察蜂阶段如算法2所示．

**算法2 基于贝叶斯网络的观察蜂算法**

**输入：当前种群** $\mathbf{POP}_C$；

**输出：更新后的种群** $\mathbf{POP}_N$；

采用锦标赛选择算法从 $\mathbf{POP}_C$ 中选取 $PS\times\theta$ 个优质食物源构建贝叶斯网络；

建立禁忌表 $Tabu=\left[\psi_1,\psi_2,\cdots,\psi_n\right]$，令 $\psi_i=0(i=1,2,\cdots,n)$；

使用轮盘赌策略选择工件操作序贝叶斯网络的节点 $N_{1,i}$ 和机器分配序贝叶斯网络的节点 $N_{1,j},(j=1,2,\cdots,m)$，令 $\pi_1^s=i,\pi_1^a=j,\psi_i=\psi_i+1$，pos=2；

Repeat

使用轮盘赌策略选择 $\boldsymbol{\pi}^s$ 网络的节点 $N_{\mathrm{pos},i}$ 和 $\boldsymbol{\pi}^a$ 网络的节点 $N_{\mathrm{pos},j}$

Repeat

将 $N_{\mathrm{pos},i}$ 与 $N_{\mathrm{pos}-1,k_1}$ 间定向弧权置0（$k_1$ 为与节点 $N_{\mathrm{pos},i}$ 的第pos层节点的个数）；

将 $N_{\mathrm{pos}-1,k_2}$ 到 $N_{\mathrm{pos},k}(k=1,2,\cdots,i-1,i+1,n)$ 定向弧权重新归一化处理；

计算 $P(N_{\mathrm{pos},k}\mid N_{\mathrm{pos}-1,k})$，继而使用轮盘赌重新选择节点 $N_{\mathrm{pos},i},\pi_{\mathrm{pos}}^s=i,\psi_i=\psi_i+1$；

Until $\psi_i\leqslant n_i$；

If 工序 $i$ 不能在机器 $j$ 上加工 then

随机选择当前操作可利用且累积负荷最小的机器 $j'$ 对工序 $i$ 进行加工，$\pi_{\mathrm{pos}}^a=j'$；

Else

$\pi_{\mathrm{pos}}^a=j,\mathrm{pos}=\mathrm{pos}+1$；

Until pos−1>PS；

$\boldsymbol{\pi}^w$ 为种群 $\mathrm{POP}_C$ 中最差的个体，采用新生成的个体替换 $\boldsymbol{\pi}^w$ 的到新种群 $\mathrm{POP}_N$；

由于优质食物源的编码在序关系和位置上存在相似的块结构，而这些相似的块结构及其编码位置对应贝叶斯网络中具有高权重的定向弧，因此使用改进的基于贝叶斯网络的概率模型可以较好的保留优质序关系，从而有更高的概率生成优质食物源，从而在加快算法的收敛速度的同时实现种群的协同进化．

### 3.5 改进侦察蜂阶段

当最优个体连续 $\eta$ 代未更新，则意味着算法可能陷入了局部最优，此时调用改进侦察蜂算法更新种群，具体方法如下．

首先对当前最优解的工件操作序列执行3次扰动操作，并采用3.2节的算法1生成机器分配序从而组成新个体，重复上述操作生成PS/2个个体来替换老种群中适应度值较差的一半．本文所设计的扰动算子如下．

（1）swap（$\boldsymbol{\pi}^s$）：随机选择序 $\boldsymbol{\pi}^s$ 中的两个位置 $a,b$，并把 $a,b$ 中间的序列逆序．

（2）insert（$\boldsymbol{\pi}^s$）：随机选择序 $\boldsymbol{\pi}^s$ 中的一个位置 $i$，并把位置 $i$ 后面的序插到序列最前面．

### 3.6 算法流程图及复杂度分析

为克服传统算法在求解SFTSP时存在的不足，本文借鉴人工蜂群算法框架的灵活性，融合了多种搜索策略和基于贝叶斯网络的概率模型，设计了一种增强人工蜂群算法．算法流程图如图5所示．

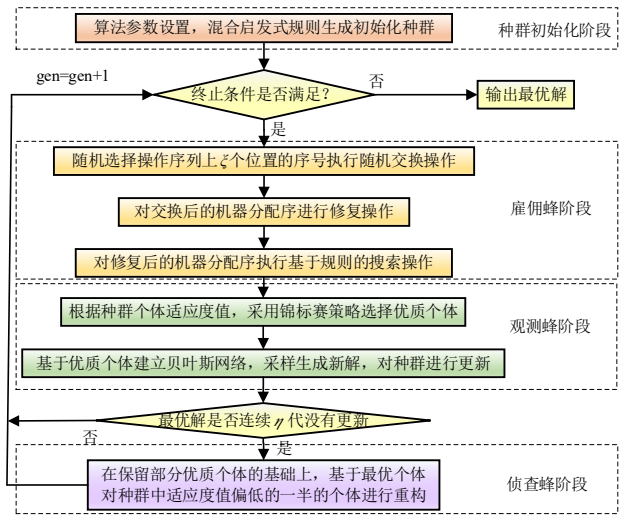


图5 EABC算法流程图

由图5可以看出，EABC的时间复杂度主要包括种群初始化阶段、雇佣蜂阶段、观察蜂阶段和侦察蜂阶段时间复杂度分析如下：

（1）种群初始化阶段的复杂度包括混合启发式算法生成解序列 $O(n\log n)$ 和计算每个个体的最大完工时间 $O(TS\times m)$，考虑种群规模，则种群初始化阶段的时间复杂度为 $O(PS\times(TS\times m+n\log n))$.

（2）雇佣蜂阶段时间复杂度为 $O(PS\times TS\times m)$.

（3）观察蜂阶段时间复杂度包括采样贝叶斯网络生成新个体 $O(TS^2+TS\times m)$ 和利用新个体替换种群中最差个体 $O(PS)$，所以观测蜂阶段时间复杂度为 $O(TS^2+TS\times m+PS)$.

（4）侦察蜂阶段时间复杂度为 $O(PS^2+PS\times TS\times m)$.

设置算法最大运行代数为 $\mathrm{gen}_{\max}$，EABC的整体时间复杂度为：

$$O(PS,TS,n,m,\mathrm{gen}_{\max})=O(PS\times(TS\times m+n\log n))+\mathrm{gen}_{\max}(O(PS^2+PS\times TS\times m)+O(TS^2+TS\times m+PS))$$

从上式可知，EABC时间复杂度的最高次项是工序总数TS和种群大小PS（实际上PS是一个较小值，本文为10），故EABC具有较低时间复杂度．

## 4　实验分析与比较

为确定算法的参数和性能,对其开展实验设计,采用Wu等[7]根据真实半导体最终测试过程提供的10组测试实例(可通过http://dalab.ie.nthu.edu.tw/news-en_content.php?id=0进行算例下载),其中包括"大数据"(LS)实例和"大范围"(WR)实例各五组.

### 4.1　参数设置

EABC的关键参数包括种群规模PS、精英个体占比$\theta$、迭代未更新次数$\eta$,考虑计算效率和解决方案质量,EABC参数决定如下:"大范围"实例参数采用PS=10、$\theta$ = 0.4、$\eta$ = 3,"大数据"实例参数采用PS=10、$\theta$ = 0.4、$\eta$ = 2.

### 4.2　EABC关键环节有效性验证

本节设计了四种ABC的变形算法与EABC进行对比实验,分别对算法中各关键环节有效性进行验证.变形算法如下:

(1)EABC_ND:解码过程采用文献[9]的方法,其余部分与EABC一致.

(2)EABC_NM:雇佣蜂阶段采用单一策略的搜索方式,其余部分与EABC一致.

(3)EABC_NB:观察蜂阶段采用锦标赛算法选择个体进行交叉操作来更新种群,其余部分与EABC一致.

(4)EABC_NS:侦察蜂阶段对种群中重复或者迭代过程中没有更新的解进行重构,其余部分与EABC一致.

设定每种算法终止时间为$n \times m \times 3$,每个测试算例均进行20次独立试验,各指标对应的最优结果用粗体表示.由表2可知,EABC在大部分问题上的测试结果都明显优于所构造的对比算法,这验证了EABC中各关键环节的有效性.

**表2**　EABC与其多种变形算法对比结果($\lambda$ = 3)

| 问题 | EABC_ND | | | EABC_NM | | | EABC_NB | | | EABC_NS | | | EABC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 规模 | BST | WST | AVG | BST | WST | AVG | BST | WST | AVG | BST | WST | AVG | BST | WST | AVG |
| LS1 | 96.0 | 110.0 | 101.1 | 99.0 | 114.0 | 103.9 | 96.0 | 112.0 | 101.2 | 95.0 | 114.0 | 105.7 | **84.0** | **105.0** | **91.3** |
| LS2 | 108.0 | 127.0 | 115.6 | 115.0 | 139.0 | 125.7 | 113.0 | 138.0 | 120.8 | 101.0 | 119.0 | 109.7 | **96.0** | **106.0** | **99.8** |
| LS3 | 96.0 | 119.0 | 107.9 | 106.0 | 125.0 | 116.6 | 104.0 | 120.0 | 111.6 | 100.0 | 114.0 | 107.9 | **91.0** | **102.0** | **95.8** |
| LS4 | 117.0 | 135.0 | 124.7 | 123.0 | 148.0 | 132.9 | 116.0 | 145.0 | 124.3 | 114.0 | 136.0 | 126.2 | **107.0** | **120.0** | **111.9** |
| LS5 | **96.0** | 121.0 | 104.6 | 98.0 | 124.0 | 110.7 | **96.0** | 114.0 | 102.9 | **96.0** | **106.0** | **100.6** | **96.0** | 107.0 | **100.6** |
| WR1 | 199.0 | 240.0 | 216.3 | 226.0 | 290.0 | 246.7 | 208.0 | 247.0 | 231.6 | 218.0 | 245.0 | 228.4 | **215.0** | **243.0** | **224.4** |
| WR2 | 186.0 | 210.0 | 197.7 | 194.0 | 244.0 | 216.2 | 191.0 | 210.0 | 201.6 | 184.0 | 208.0 | 196.8 | **173.0** | **194.0** | **183.1** |
| WR3 | 200.0 | 235.0 | 218.0 | 208.0 | 262.0 | 231.3 | 197.0 | 223.0 | 213.8 | 194.0 | 224.0 | 209.2 | **193.0** | **226.0** | **203.8** |
| WR4 | 193.0 | 237.0 | 210.0 | 202.0 | 238.0 | 222.8 | 195.0 | 222.0 | 206.2 | 189.0 | 211.0 | 200.9 | **183.0** | **192.0** | **187.0** |
| WR5 | 203.0 | 242.0 | 218.9 | 213.0 | 273.0 | 241.4 | 194.0 | 233.0 | 211.2 | 198.0 | 235.0 | 217.2 | **189.0** | **212.0** | **198.9** |
| 均值 | 149.3 | 177.6 | 161.5 | 158.4 | 195.7 | 174.8 | 151.0 | 176.4 | 162.5 | 148.8 | 171.2 | 160.2 | **142.7** | **162.1** | **149.8** |

**表3**　EABC与NFOA、KMEA和CCIWO的比较($\lambda$ = 3)

| 问题 | NFOA | | | KMEA | | | CCIWO | | | EABC | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 规模 | BST | WST | AVG | BST | WST | AVG | BST | WST | AVG | BST | WST | AVG |
| LS1 | 103.0 | 127.0 | 115.9 | 95.0. | 106.0 | 101.1 | 89.0. | **102.0** | 94.0 | **84.0** | 105.0 | **91.3** |
| LS2 | 110.0 | 128.0 | 116.8 | 99.0 | 112.0 | 106.2 | 98.0 | 109.0 | 101.9 | **96.0** | **106.0** | **99.8** |
| LS3 | 105.0 | 118.0 | 110.6 | 96.0 | 104.0 | 99.9 | 92.0 | 104.0 | 97.0 | **91.0** | **102.0.** | **95.8** |
| LS4 | 122.0 | 131.0 | 125.6 | 116.0 | 125.0 | 119.1 | 109.0 | 121.0 | 112.5 | **107.0** | **120.0** | **111.9** |
| LS5 | 116.0 | 126.0 | 120.4 | 106.0 | 116.0 | 110.0 | 98.0 | 109.0 | 102.8 | **96.0** | **107.0** | **100.6** |
| WR1 | 230.0 | 264.0 | 247.5 | 227.0 | 257.0 | 238.9 | **215.0** | **238.0.** | 224.8 | **215.0** | 243.0 | **224.4** |
| WR2 | 192.0 | 240.0 | 217.7 | 180.0 | 210.0 | 191.2 | 174.0 | 195.0 | **180.3** | **173.0** | **194.0.** | 183.1 |
| WR3 | 216.0 | 243.0 | 225.4 | 208.0 | 236.0 | 217.5 | **193.0** | 227.0 | 205.4 | **193.0** | **226.0** | **203.8** |
| WR4 | 185.0 | 222.0 | 201.2 | 186.0 | 198.0 | 191.1 | 184.0 | 198.0 | 189.1 | **183.0** | **192.0** | **187.1** |
| WR5 | 214.0 | 235.0 | 226.3 | 197.0 | 219.0 | 205.2 | 193.0 | 219.0 | 202.3 | **189.0** | **212.0** | **198.9** |
| 均值 | 159.3 | 183.4 | 170.7 | 151.0 | 168.3 | 158.1 | 144.5 | 162.2 | 151.1 | **142.7** | **162.1** | **149.9** |

### 4.3　EABC与其他算法对比

为进一步验证EABC的性能,将EABC和求解SFTSP的有效算法NFOA[7]、KMEA[17]和CCIWO[8]进行比较.设定每种算法终止时间为$n \times m \times \lambda$,$\lambda$取1、2、3三水平,每个测试算例均进行20次独立试验,各指标对应的最优结果用粗体表示.由表3($\lambda$ = 3)可知,

EABC在绝大部分算例上的测试结果明显优于对比算法.此外,为直观展示EABC与各对比算法实验结果的离散程度和对称性,绘制实例LS3、WR1和WR4的箱线图,由图6可以看出,EABC的整体效果要优于其他三种对比算法,且随运行时间的增大,大部分实例的最优解都有所改进.此外,根据λ在1、2和3三种取值下每种算法的占优比进行方差分析(ANOVA),来进一步验证各算法组内差异.图7显示了四种算法在不同终止运行时间下的均值变化线及95%置信度下的Tukey's HSD检验的置信区间,可以看出,EABC的占优比在均值水平上与NFOA、KMEA有显著差异,且优于CCIWO.

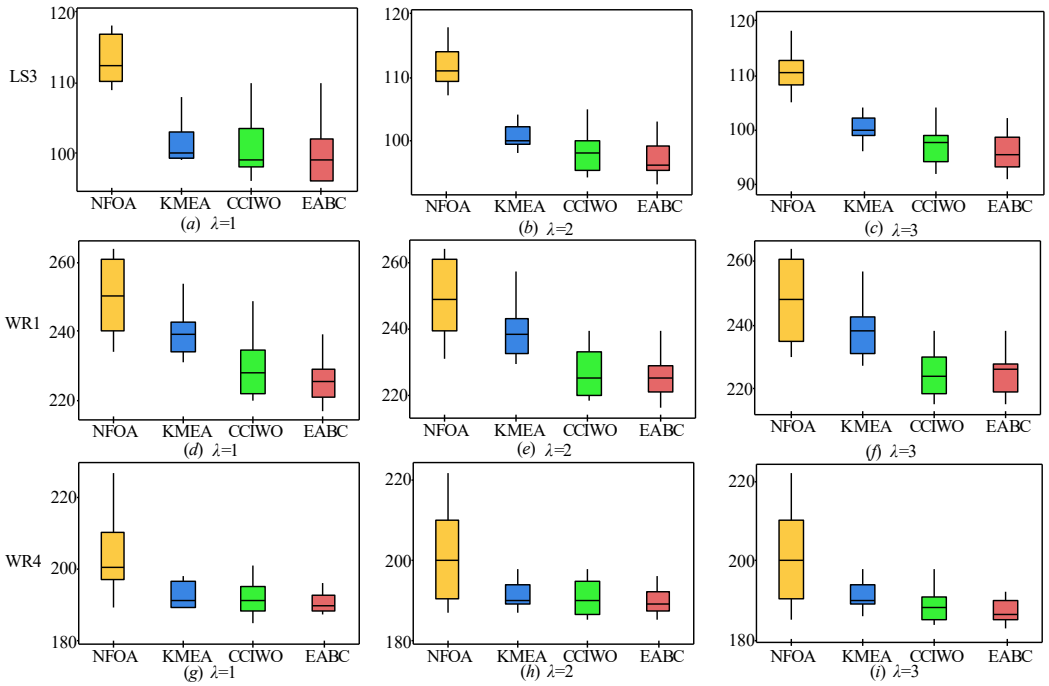


图6 实例LS3、WR1和WR4的实验结果箱线图

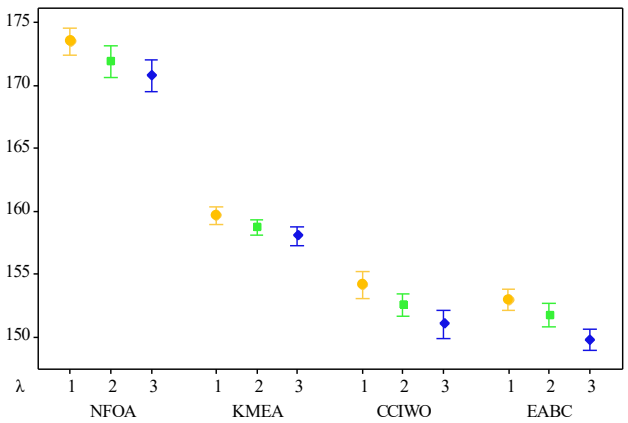


图7 EABC与对比算法的方差分析区间图

## 5 结论

本文针对半导体最终测试调度问题(SFTSP),提出了一种增强人工蜂群算法(EABC)进行求解.EABC具有如下特点:(1)采用混合启发式方法初始化种群,并结合前插式解码策略对问题解码过程进行优化,有利于保证初始解的质量和分散性.(2)设计包含多种搜索策略的雇佣蜂阶段来保证算法对解空间的搜索深度.(3)引入基于贝叶斯网络的概率模型来学习和积累种群中的优质信息(4)为了防止算法陷入局部最优的快速下降陷阱,设计用于种群重构的改进侦察蜂阶段,来进一步提升算法的搜索性能.

## 参考文献

[1] 朱泓达.中国半导体硅的现状与发展趋势[J].数字通信世界,2020,19(6):268-278.
ZHU H D. The status quo and development trend of semiconductor silicon in China[J].Digital Communication Circle,2020,19(6):268-278.(in Chinese)

[2] UZSOY R, LEE C Y, MARTIN-VEGA L A. Scheduling semiconductor test operations: Minimizing maximum lateness and number of tardy jobs on a single machine[J]. Naval Research Logs, 1992, 39(3): 369-388.

[3] UZSOY R, CHURCH L K, OVACIK I M, et al. Performance evaluation of dispatching rules for semiconductor testing operations[J]. Journal of Electronics Manufactur-

ing, 2012, 3(2): 95 – 105.
[4] WU J Z, CHIEN C F. Modeling semiconductor testing job scheduling and dynamic testing machine configuration[J]. Expert Systems with Applications, 2008, 35(1-2): 485 – 496.
[5] WU J Z, HAO X C, CHIEN C F, et al. A novel bi-vector encoding genetic algorithm for the simultaneous multiple resources scheduling problem [J]. Journal of Intelligent Manufacturing, 2012, 23(6): 2255 – 2270.
[6] HAO X C, et al. The cooperative estimation of distribution algorithm: a novel approach for semiconductor final test scheduling problems[J]. Journal of Intelligent Manufacturing, 2014, 25(5): 867 – 879.
[7] ZHENG X L, WANG L, WANG S Y. A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem [J]. Knowledge-Based Systems, 2014, 57(2): 95 – 103.
[8] SANG H Y, DUAN P Y, LI J Q. An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem[J]. Swarm and Evolutionary Computation, 2018, 38(1): 1 – 12.
[9] LU S J, LIU X B, PEI J, et al. A hybrid ABC-TS algorithm for the unrelated parallel-batching machines scheduling problem with deteriorating jobs and maintenance activity [J]. Applied Soft Computing, 2018, 66(1): 168 – 182.
[10] LI J Q, SONG M X, WANG L, et al. Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs [J]. IEEE Transactions on Cybernetics, 2020, 50(6): 1 – 15.
[11] GAO K Z, SUGANTHAN P N, CHUA T J, et al. A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion [J]. Expert Systems with Applications, 2015, 42(21): 7652 – 7663.
[12] GONG G L, CHIONG R, DENG Q W, et al. A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility [J]. International Journal of Production Research, 2020, 58(14): 1 – 15.
[13] LI Y, HUANG W, WU R, et al. An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem[J]. Applied Soft Computing, 2020, 95(5): 1 – 14.
[14] BANHARNSAKUN A, ACHALAKUL T, SIRINAOVAKUL B. The best-so-far selection in artificial bee colony algorithm [J]. Applied Soft Computing Journal, 2010, 11(2): 2888 – 2901.
[15] LI H, LI X, GAO L. A discrete artificial bee colony algorithm for the distributed heterogeneous no-wait flowshop scheduling problem [J]. Applied Soft Computing, 2021, 100(1): 1 – 19.
[16] 赵博选, 高建, 付颖斌, 等. 求解柔性作业车间调度问题的多策略融合 Pareto 人工蜂群算法[J]. 系统工程理论与实践, 2019, 39(5): 1225 – 1235.
ZHAO BOXUAN, GAO JIAN, FU YINGBIN, et al. Multi-strategy fusion Pareto artificial bee colony algorithm for solving flexible job shop scheduling problems [J]. Systems Engineering Theory and Practice, 2019, 39(5): 1225 – 1235.(in Chinese)
[17] WANG S, WANG L. A knowledge-based multi-agent evolutionary algorithm for semiconductor final testing scheduling problem [J]. Knowledge-Based Systems, 2015, 84(8): 1 – 9.

## 作者简介

**吕　阳**　男,1996年生于曲靖.现为昆明理工大学大学信息工程与自动化学院硕士研究生.主要研究方向为智能算法与优化调度.
E-mail: 726564418@qq.com

**钱　斌(通讯作者)**　男,1976年出生,教授,博士研究生导师.主要研究方向为优化调度理论与方法.
E-mail:bin.qian@vip.163.com

**胡　蓉**　女,1973年出生,副教授,硕士研究生导师.主要研究方向为优化方法和决策支持系统.
E-mail:ronghu@vip.163.com

**张梓琪**　男,1989年生于云南曲靖.现为昆明理工大学大学信息工程与自动化学院博士研究生.主要研究方向为智能算法与优化调度.
E-mail: 768894018@qq.com