文章编号: 1000−8152(2011)12−1813−08

# 基于蚁群−混合蛙跳算法的贴片机贴装顺序优化

陈铁梅[1,2], 罗家祥[2], 胡跃明[2]

(1. 广东商学院 信息学院, 广东 广州 510320;

2. 华南理工大学 自动化科学与工程学院 精密电子制造装备教育部工程研究中心, 广东 广州 510640)

**摘要**: 针对喂料器的位置确定的条件下, 研究拱架式贴片机的元器件贴装顺序优化问题. 建立了新的拱架式贴片机贴装顺序的数学模型. 针对问题的路径寻优特点, 把混合蛙跳算法与蚁群算法相融合, 实现对贴片机的元件贴装顺序优化问题的求解. 在算法中提出了适应于贴片机实际贴装情况的分段启发函数、分段信息素以及信息素的分段更新策略等多种改进方法. 为验证算法有效性, 以20块实际生产的PCB为实例进行了测试. 实验结果表明, 算法具有较好的求解精度和全局搜索能力, 与文献中的单一混合蛙跳算法相比, 平均效率提高了7.89%; 与蚁群算法相比, 平均效率提高了3.79%.

**关键词**: 贴装顺序优化; 蚁群算法; 混合蛙跳算法

**中图分类号**: TP202.7 **文献标识码**: A

# Mounting sequence optimization on surface mounting machine using ant-colony algorithm and shuffled frog-leaping algorithm

CHEN Tie-mei[1,2], LUO Jia-xiang[2], HU Yue-ming[2]

(1. Information Science School, Guangdong University of Business Studies, Guangzhou Guangdong 510320, China;
2. Engineering Research Center for Precision Electronic Manufacturing Equipments of Ministry of Education,
College of Automation Science and Technology, South China University of Technology, Guangzhou Guangdong 510640, China)

**Abstract:** The component mounting sequence optimization of the surface mounting machine is considered under the condition that the feeder allocations are known. A new mathematical model of mounting sequence is specifically built for arch surface mounting machines. Based on the characteristics of searching for the optimal path, a new hybrid algorithm of ant-colony algorithm merged with the shuffled frog-leaping algorithm is proposed to solve the problem. According to the actual mounting situation, a few improved methods are proposed in the algorithm, such as the segmented heuristic function, the segmented pheromone, and the pheromone update strategy. To verify the efficiency of the algorithm, component mounting experiments of 20 printed-circuit-boards(PCBs) are tested. The results show the algorithm has higher accuracy in solving the problem, and in searching the optimal path. It provides an improvement in average efficiency 7.89% over the single shuffled frog-leaping algorithm, and 3.79% over the single ant-colony algorithm.

**Key words:** component mounting sequence optimization; ant-colony algorithm; shuffled frog-leaping algorithm

## 1 引言(Introduction)

贴片机是将各种片式元器件贴装到PCB或将裸芯片贴到封装基板的工作母机. 贴片机的优化一般分为元器件贴装顺序确定条件下的喂料器分配问题和喂料器固定的元器件的贴装顺序优化问题两大问题. 前者通常认为是一个二次分配问题(quadratic assignment problem, QAP)[1], 后者是带约束的TSP (travelling salesman problem)问题[1,2]. 本文主要研究喂料器固定的前提下元器件的贴装顺序优化问题.

在对贴片机贴装顺序优化问题的研究中, 启发式算法[1,3]、遗传算法[4~7]、蚁群算法[8]、微粒群算法[9]、混合蛙跳算法[10,11]等智能优化算法的应用, 均能在一定的时间内取得较好的贴片机贴装顺序优化效果. 但同时单个算法又不可避免地存在一定的缺陷, 遗传算法是个体之间缺乏信息的交流和传递, 算法求解到一定范围时最优解收敛速度降低, 蚁群算法由于个体之间不断进行信息的交流, 信息素的正反馈作用容易陷入局部最优等. 混合蛙跳算法具有概念简单, 调整的参数少, 计算速度快, 全局搜索寻优能力强等优点, 但存在对系统中反馈信息利用不够, 大量迭代活动, 求精度解效率低等缺点[12].

因此本文针对拱架式贴片机, 在喂料器位置确定的情况下, 以PCB板上所有的元器件吸取和贴装完毕时贴片头移动的总距离为优化目标, 建立了

收稿日期: 2010−07−30; 收修改稿日期: 2011−05−16.

基金项目: 国家自然科学基金资助项目(60835001, 60804053, 61105081); 教育部重点科研基金资助项目(200805611065); 广东省科技攻关重大资助项目(912220500017).

一个新的拱架式贴片机贴装顺序数学模型, 与文献[8, 10, 11]中模型相比, 新模型更加全面的反映了贴片机的贴装特点和更具体的描述了贴片机的贴装过程. 该模型属于TSP模型的扩展, 研究问题具有寻找最优路径的特点. 利用蚁群算法与混合蛙跳算法的混合算法对贴片机的元件贴装顺序优化问题进行求解, 蚁群算法中提出了一种适应于贴片机实际贴装情况的分段启发函数和分段信息素以及信息素的组更新策略. 为了提高蚁群算法的分散搜索能力, 将蚂蚁搜索的结果作为混合蛙跳算法初始蛙群, 进一步进行搜索, 有效的避免蚁群算法易陷入局部最优. 最后以20块实际生产的PCB为实例进行了仿真测试, 结果表明, 蚁群–混合蛙跳算法比混合蛙跳算法和蚁群算法更高的求解精度和更强的全局搜索能力.

## 2 拱架式贴片机工作原理和模型(Principle and model of arch surface mounting machines)

多贴装头拱架式贴片机结构简图如图1所示. 它主要包括喂料器、贴装头、吸嘴和PCB等部分. 装载元器的喂料器固定在两边的喂料槽上, 每个喂料器占据一个或多个喂料槽, 每个喂料器上只能装载同一种元器件, 所有的贴片头固定安装在拱架上, 所有贴片头的移动是通过拱架的移动来实现的. 元器件的取贴等动作都是由装在贴片头上的吸嘴来完成. 由于贴片头之间的距离是供料槽之间距离的整数倍, 所以贴片头可以同时吸取多个元器件.

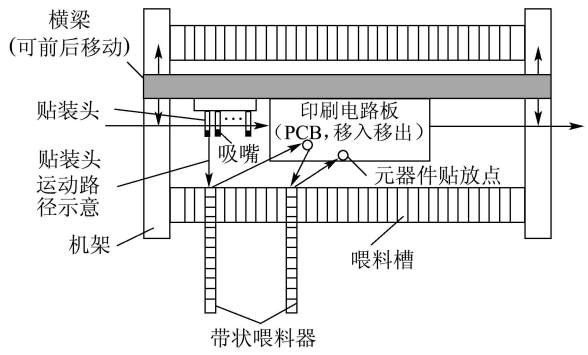


图 1 拱架式贴片机结构图

Fig. 1 Frame diagram of arch surface mounting machines

贴片过程大致如下: 首先贴装头移动至元器件喂料器位置依次或同时拾取$H$个($H$为贴装头吸嘴的个数)元器件, 经过视觉检测校正, 贴装头移动到第1个器件的装配位置将器件1贴装, 再移动至第2个器件贴装的位置将器件2贴装, 依此直至$H$个元器件贴装完毕之后, 这样一个取贴操作过程被称为一个取贴循环, 如此循环取贴, 直到所有元器件取贴完毕.

因为整个贴片过程较复杂, 本文假设同一吸嘴可以吸取不同类型的元器件, 每次吸嘴都能正确无误的吸取元器件. 设有元器件的贴装序列$X = \{x_{[1]}, x_{[2]}, \cdots, x_{[3]}, \cdots, x_{[CY]}\}$, 其中: $x_{[i]}$表示第$i$个顺序的元器件以及该元器件在PCB板上的坐标, $CY$为元器件的总个数. 由于喂料槽的位置是确定的, 因此每个贴片序列$X$对应一个喂料槽顺序集合$R = \{r_{[1]}, r_{[2]}, \cdots, r_{[i]}, \cdots, r_{[CY]}\}$, 其中$r_{[i]}$表示贴装序列中第$i$个元器件相对应的喂料器以及喂料器的坐标. 设$H$为贴片头吸嘴的个数, 设$CY$为$H$的倍数, $C = \lceil \dfrac{CY}{H} \rceil$ 为取贴循环的总数, 因此贴片机贴装路径优化问题归纳为: 寻找贴装顺序集合$X$使式(1)的目标函数最小, 即把贴片机贴片顺序优化转化为所有的元器件吸取和贴装完毕时贴片头移动的总距离最小.

$$\begin{aligned} &Y(X) = \\ &\min\Big(\sum_{k=1}^{C-1} d_1(x_{[H\times k]}, r_{[H\times(k+1)]}) + \\ &\sum_{i=1}^{CY-1} d_2(r_{[i]}, r_{[(i+1)]}) - \sum_{k=1}^{C} d_2(r_{[H\times k]}, r_{[H\times(k+1)]}) + \\ &\sum_{k=1}^{C} d_3(r_{[H\times k]}, x_{[H\times(k-3)]}) + \sum_{i=1}^{CY-1} d_4(x_{[i]}, x_{[(i+1)]}) - \\ &\sum_{k=1}^{C-1} d_4(x_{[H\times k]}, x_{[H\times(k+1)]})\Big). \end{aligned} \tag{1}$$

$d_1(x_{[H\times k]}, r_{[H\times(k+1)]})$表示从取贴循环$k$的最后一个贴装位置到取贴循环$k+1$的初始元器件的所对应喂料槽的欧式距离.

$d_2(r_{[i]}, r_{[(i+1)]})$表示贴片头的当前吸嘴从第$i$的元器件所对应的喂料槽吸取该元器件到相邻的下一个吸嘴移动到序列中第$i+1$的元器件所对应的喂料槽吸取该元器件时贴片头移动的距离, $d_2(r_{[H\times k]}, r_{[H\times(k+1)]})$为贴片头的当前吸嘴从第$k$个循环的最后一个元器件对应的喂料槽吸取该元器件到相邻的下一个吸嘴从第$k+1$个取贴循环的第1个元器件所对应喂料槽吸取该元器件时贴片头的移动距离.

$d_3(r_{[H\times k]}, r_{[H\times(k-3)]})$表示贴片头离开第$k$次循环最后一个吸取元器件的喂料槽的位置移动本次循环第1个贴装元器件在PCB位置的欧式距离.

$d_4(x_{[i]}, x_{[(i+1)]})$表示相邻元器件在PCB板上的欧式距离, $d_4(x_{[H\times k]}, x_{[H\times(k+1)]})$为第$k$个循环取贴循环中的最后一个元器件到第$k+1$个取贴循环中元器件的欧式距离.

本文建立的新模型克服了以往文献[8, 10, 11]中模型的不足: 文献[8]中的模型是在假定贴片机的元器件拾取都在同一位置的这种简化条件的基础上建立的; 文献[10, 11]所建立的模型缺乏对贴片头各种

移动距离的具体描述, 因此新模型更加全面的反映了贴片机的贴装特点和更具体的描述了贴片机的贴装过程.

## 3 算法(Algorithm)

### 3.1 改进的蚁群算法(Improved ant colony algorithm)

蚁群算法中蚂蚁个体通过信息素进行信息交流和传递, 形成一种正反馈机制, 使蚁群最终可发现最短路径.

将蚁群算法用于求解贴片机贴装顺序优化问题, 关键在于蚂蚁的路径的构建过程以及信息素的更新. 图2为单只蚂蚁的路径构造过程. 首先蚂蚁随机选择一个元件作为路径的起点, 蚂蚁根据不同的状态转移策略选择循环内的元器件和下一个循环的第1个元器件, 直至所有的元器件都选择完, 形成一条路径, 即获得贴片顺序优化的一个解. 根据路径的长度, 更新信息素. 经过不断的重复迭代, 路程最短的路径即为蚂蚁搜索得到的元器件优化方案.

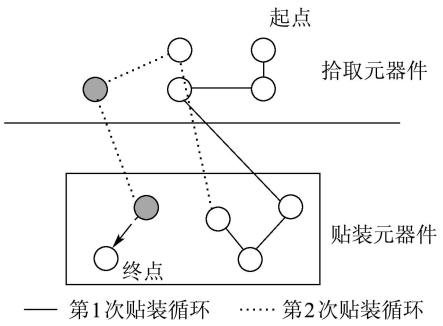


图 2 单只蚂蚁的路径构建图

Fig. 2 Route construction of per ant

贴片机的贴装过程核心是拾取过程和贴片过程, 贴片头的移动可划分为在喂料器上、在喂料器和PCB板之间以及在PCB上的移动. 由于喂料器的位置是确定的, 贴片头在喂料器上的移动顺序随着元器件贴装顺序的确定而确定. 因此蚂蚁选择路径主要与后面两种移动有关, 蚂蚁的路径构建可分成2个阶段, 第1个阶段当元器件在同一个贴装循环时, 根据信息素和启发函数进行路径选择, 这种启发函数和信息素主要与同一拾取循环中元器件在PCB上的欧式距离相关. 当元器件不在同一个取贴循环中, 此时进行路径构建仍采用第1阶段相同的信息素和启发函数不能很好的反映贴片机的实际贴装过程, 因此第2阶段当元器件不在同一个取贴循环时, 根据第2种信息素和启发函数进行路径选择, 这种启发函数和信息素主要与元器在PCB板上的位置与喂料器之间的欧式距离相关. 因此在蚂蚁的路径构建过程中, 为了蚂蚁能更准确的选择符合贴片机实际贴装情况的贴片顺序, 提出了分段启发函数、分段信息素等多种改进方法来改进蚂蚁的状态转移策略, 同时相应地在信息素的更新中, 采了信息素的分段更新策略, 即第1种信息素迄今最优路径组更新策略和迄今最优路径更新策略以及第2种信息素的二次更新策略.

#### 3.1.1 状态转移策略(Statue transition strategy)

在蚂蚁算法中, 位于元器件$i$的蚂蚁$k$根据伪随机比例规则选择元器件$j$, 其规则如下:

$$j=\begin{cases}\arg\max([\tau_{il}(t)]^{\alpha}\times[\eta_{il}(t)]^{\beta})q\leqslant q_0,\ l\in SS,\\ P_{ij}, \qquad\qquad\qquad\qquad\qquad 其他,\end{cases} \tag{2}$$

其中: $q\in U(0,1)$, $q_0\in(0,1)$是预先设置阈值. 这种规则表明, 蚂蚁选择当前可能的最优移动方式的概率是$q_0$, 这种最优的移动方式是根据信息素的积累量和启发式信息值求出的, 即通过式(3)求出. 同时, 蚂蚁以$(1-q_0)$的概率探索其他区域.

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum\limits_{l\in SS}[\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}},\ l\in SS,\\ 0, \qquad\qquad\qquad\qquad 其他.\end{cases} \tag{3}$$

$P_{ij}^{k}(t)$为$t$时刻第$k$只蚂蚁从元件$i$选择元件$j$的状态转移概率, 其中: $\alpha$为信息启发因子, $\beta$为期望启发式因子, $SS$ 为待选择的元器件的集合. $\tau_{ij}(t)$为信息素, 分两种情况, 即元器件$i$和元器件$j$在同一取贴循环时元件$i$到元件$j$的信息素, 记为$\tau 1_{ij}(t)$, 以及元器件$i$和元器件$j$不在同一取贴循环中, 元件$i$是当前取贴循环的最后元件, 元件$j$是下一个取贴循环的第1个元件时的信息素记为$\tau 2_{ij}(t)$. $\eta_{ij}(t)$为启发函数, 根据贴片机的取贴循环的贴装路径特性, 对启发函数进行了改进, 当$t$时刻蚂蚁元器件$i$和元器件$j$在同一取贴循环中, 其启发函数为$\eta 1_{ij}(t)$:

$$\eta 1_{ij}(t)=\frac{C_1}{\mathrm{Dis}1_{ij}},\ i,j=1,2,\cdots,CY,\ i\neq j, \tag{4}$$

其中: Dis1为PCB板各元器件的欧式距离矩阵, $\mathrm{Dis}1_{ij}$为元器件$i$和元器件$j$在PCB板上欧式距离. $C_1$为常数. 当$t$时刻蚂蚁元器件$i$和元器件$j$在同一取贴循环中, 元件$i$是当前取贴循环的最后元件, 元件$j$是下一个取贴循环的第1个元件时, 启发函数$\eta 2_{ij}(t)$:

$$\eta 2_{ij}(t)=\frac{C_2}{\mathrm{Dis}2_{ij}},\ i,j=1,2,\cdots,CY,\ i\neq j, \tag{5}$$

其中: Dis2为为元器件与下一元器件所在的喂料器的欧式距离以及该喂料器与该元器件在PCB上的位置的欧式距离之和的矩阵, $\mathrm{Dis}2_{ij}$为元器件$i$与元器件$j$所在的喂料器的欧式距离以及元器件$j$所在的喂料器与元器件$j$在PCB板上的位置的欧式距离之和, $C_2$为常数.

### 3.1.2 分段的信息素更新规则(Segmented pheromone update rules)

在信息素更新规则上, 由于有两种不同的信息素, 信息素有不同的更新规则: 即第1种信息素的迄今最优路径的组更新和迄今最优路径更新策略以及第2种信息素二次更新策略.

元器件$i$元器件$j$在同一取贴循环时, 首先采用迄今最优路径的组更新原则, 当在迄今最优路径中每个取贴循环结束时, 减少所有路径上的信息素, 同时增加迄今最优路径上的当前取贴循环路径上的信息素.

$$\tau_{ij}(t)=(1-\rho)\tau_{ij}^{t-1}+\Delta\tau_{ij}^{t}, \tag{6}$$

$$\tau_{ij}^{t}=\begin{cases}\dfrac{Q_1}{L_1}, & \text{若迄今最优路径上本次取贴}\\ & \text{循环包括元器件}i\text{和}j,\\ 0, & \text{否则,}\end{cases} \tag{7}$$

其中: $\rho$为信息速挥发系数, $L_1$为迄今最优路上本次取贴循环的路径长度, $Q_1$为信息素的强度.

然后根据式(6)(8)对第1种信息素进行迄今最优路径的更新策略.

$$\tau_{ij}^{t}=\begin{cases}\dfrac{Q_2}{L_2}, & \text{若迄今最优路径上包括元器件}i\text{和}j,\\ 0, & \text{否则,}\end{cases} \tag{8}$$

其中$L_2$为迄今最优路上的路径长度.

第1种信息素的迄今最优路径组更新策略能更好的把每一个取贴循环的特性反映在信息素上, 更加符合贴片机的贴装过程.

当元器件$i$和元器件$j$不在同一取贴循环中, 元件$i$是当前取贴循环的最后元件, 元件$j$是下一个取贴循环的第1个元件时, 采取二次更新信息素的办法. 在每只蚂蚁构建路径时, 当当前取贴循环结束, 下一个取贴循环选择好开始元器件时, 立即采取式(6)(9)(10)进行第1次更新:

$$\Delta\tau_{ij}^{t}=\sum_{k=1}^{\text{Pant}}\Delta\tau_{ij}^{k\,t}. \tag{9}$$

$$\tau_{ij}^{k\,t}=\begin{cases}\dfrac{Q_3}{L_{ij}^{k}}, & \text{第}k\text{只蚂蚁本次取贴循环最后}\\ & \text{元件为}i\text{和下一次取贴循环}\\ & \text{中第1个元器件为元器件}j;\\ 0, & \text{否则,}\end{cases} \tag{10}$$

其中: $L_{ij}^{k}$为$k$只蚂蚁在本次取贴循环的最后一个元器件$i$在PCB板上的位置到下一个取贴循环中第1个元器件$j$的喂料器位置的欧式距离以及元件$j$的喂料器到元件$j$在PCB上的位置的欧式距离的和, Pant表示蚂蚁的总个数, $Q_3$为信息素的强度.

当迄今最优路径产生后, 对第2种信息素用式(6)和式(11)进行第2次更新:

$$\tau_{ij}^{t}=\begin{cases}\dfrac{Q_4}{L_{ij}^{k}}, & \text{若迄今最优路径上本次取贴}\\ & \text{循环的最后元件为}i\text{和下一次}\\ & \text{取贴循环中第1个元件}j,\\ 0, & \text{否则.}\end{cases} \tag{11}$$

其中: $L_{ij}^{k}$为迄今最优路径本次循环的最后一个元器件$i$在PCB板上的位置到下一个取贴循环中第1个元器件$j$的喂料器位置的欧式距离以及元件$j$的喂料器到元件$j$在PCB上的位置的欧式距离的和, $Q_4$为信息素的强度.

## 3.2 引入混合蛙跳算法SFLA(Introduction of shuffled frog leaping algorithm)

混合蛙跳算法(SFLA)是一种受自然生物模仿启示而产生的基于群体的协同搜索方法, 有高效的计算性能和优良的全局搜索能力, 主要包括局部搜索和全局信息交换[12]. 由于全局信息交换和局部深度搜索的平衡策略使得SFLA具有能够跳出局部极值点、向着全局最优的方向进行的优点[12], 把混合蛙跳算法的局部搜索和全局信息交换引入到蚁群算法中, 从而扩大蚁群的搜索空间, 增加蚂蚁的全局搜索能力.

因此在蚁群算法的每次迭代中, Pant只蚂蚁根据状态转移策略和信息素更新策略构造Pant个贴片机路径优化方案, Pant个优化方案作为SFLA的Pant只蛙组成SFLA的初始群体, 采用参考文献[10, 11]的模因内三角概率混合蛙跳算法进行局部迭代和混合迭代, 得到全局最优蛙$X_{\mathrm{g}}$.

## 3.3 蚁群–混合蛙跳算法具体实现(Implement of ant colony and shuffled frog leaping algorithm)

考虑到蚁群算法的的正反馈作用, 容易收敛到局部最优, 而混合蛙跳算法的局部搜索和全局信息交换能扩大蚂蚁的全局搜索能力, 把混合蛙跳算法引入蚁群算法中, 主要包括参数的初始化、蚂蚁的路径构建、混合蛙跳算法、结果输出和信息素更新等4大步骤:

**Step 1** 参数的初始化.

**Step 1.1** 初始化蚂蚁的数目Pant、混合蛙跳算法蛙群个数$Q$、元件的个数$CY$、贴片头吸嘴个数$H$、两种信息素的初值、随机选择概率$q_0$、信息素挥发系数$\rho$、信息素的强度$Q_1,Q_2,Q_3,Q_4$、信息启发因子$\alpha$、期望启发式因子$\beta$、贴片头之间的距离、元器件的喂料器位置、元器件在PCB板上的位

置坐标值、蚂蚁的迭代最大次数ECHO、蛙跳算法的模因组数$m$、每个模因组内的蛙的个数$n$、混合蛙跳算法的局部迭代次数$i_{\rm part}$、混合迭代次数代次数$i_{\rm global}$等参数;

**Step 1.2**　计算所距离矩阵Dis1, Dis2从而根据式(4)(5)计算两种启发函数, 其中: $C_1=1,\ C_2=100$.

**Step 2**　蚂蚁的路径构建.

**Step 2.1**　计算$TN1_{ij}=\tau 1_{ij}^{\alpha}\times\eta 1_{ij}^{\beta},\ TN2_{ij}=\tau 1_{ij}^{\alpha}\times\eta 1_{ij}^{\beta}$的值, 其中$i,j=1,2,\cdots,CY,\ i\neq j$;

**Step 2.2**　为蚂蚁随机选取初始位置, 设该只蚂蚁经过的城市个数为$s=1$, 随机产生$q\in[0,1]$;

**Step 2.3**　当$s\mod H\neq 0$时, 根据$TN1_{ij}$和式(2)(3)为该蚂蚁选择下一个元器件; 否则, 则根据$TN2_{ij}$和状态选择策略式(2)(3)为蚂蚁选择下一个取贴循环的第1个元件并根据式(6)(9)和式(10)对信息素$\tau 2_{ij}$进行第1次更新;

**Step 2.4**　$s=s+1$, 判断$s$是否等于$CY-1$, 如果是, 则转至下一步, 否则转Step 2.3;

**Step 2.5**　判断所有蚂蚁是否已完成路径构建, 如果是, 则转入混合蛙跳算法, 否则, 转Step 2.2.

**Step 3**　引入混合蛙跳算法.

利用Step 2得到的Pant只蚂蚁的路径优化方案作为混合蛙跳算法的初始蛙群$X_1,X_2,\cdots,X_i,\cdots,X_{\rm Pant}$, 采用3.2节中的混合蛙跳算法局部迭代$i_{\rm part}$次和混合迭代$i_{\rm global}$次, 得到全局最优蛙$X_{\rm g}$.

**Step 4**　进行信息素更新和结果输出.

**Step 4.1**　输出最优蛙和最好目标函数值, 最优蛙即为当代最优元器件贴装顺序; 求出迄今最优解;

**Step 4.2**　对迄今最优解按照式(6)~(8)进行信息素$\tau 1_{ij}$更新和式(6)(11)对信息素$\tau 2_{ij}$进行第2次更新;

**Step 4.3**　判断蚂蚁算法是否终止, 若达到蚂蚁迭代次数, 则终止蚂蚁算法输出最优解, 否则, 转Step 2继续循环.

### 3.4　算法分析(Analysis of algorithm)

#### 3.4.1　计算复杂度分析(Computational complexity analysis)

由文献[13]可知, 基本蚁群算法的时间复杂度为$\mathrm{O}(\mathrm{ECHO}\cdot CY^2\cdot\mathrm{Pant})$. 算法中采用了分段信息素和启发函数等改进措施, 不会改变蚁群算法的时间复杂度. 混合蛙跳算法的时间复杂度主要取决于局部迭代的时间复杂度, 因此混合蛙跳算法的时间复杂度为$\mathrm{O}(\mathrm{ECHO}\cdot i_{\rm global}\cdot i_{\rm part}\cdot CY^2\cdot Q)$. 文中融合算法的最坏时间复杂度为

$$T(CY)=\mathrm{O}(\mathrm{ECHO}\cdot CY^2\cdot\mathrm{Pant})+\mathrm{O}(\mathrm{ECHO}\cdot i_{\rm global}\cdot i_{\rm part}\cdot CY^2\cdot Q).$$

#### 3.4.2　收敛性分析(Convergence analysis)

本文提出的算法中引入了混合蛙跳算法, 通过混合蛙跳算法的局部搜索和全局信息交换来不断改进蚂蚁的解, 返回最优解并用它来更新信息素. 事实上, 蚁群算法收敛性证明的有效性仅依赖于构建解的方式, 而与是否使用混合蛙跳算法把这些解优化为局部最优解无关[14]. 根据文献[14]已有的3个命题和2个定理, 可证明本文所构建蚁群算法以概率1收敛于全局最优解, 从而本文提出的蚁群–混合蛙跳算法以概率1收敛于全局最优解.

**命题 1**　任意一个信息素$\tau_{ij}$都满足

$$\lim_{\theta\to\infty}\tau_{ij}(\theta)\leqslant\tau_{\max}=\frac{q_f(X^*)}{\rho},$$

其中: $X^*$为信息素的最大增量, $\theta$为迭代次数; 即$\tau_{\max}$无论如何都会受到信息素蒸发的限制, 它的取值是有界限的.

**命题 2**　一旦找到一个最优解$x^*$, 则有下式成立:

$$\forall(i,j)\in X^*,\ \lim_{\theta\to\infty}\tau_{ij}^*(\theta)=\tau_{\max}=\frac{q_f(X^*)}{\rho},$$

其中$\tau_{ij}^*$是边$(i,j)\in X^*$上的信息素.

**命题 3**　在蚂蚁的搜索过程一旦找到一个最优解$x^*$, 对任意$\tau_{ij}(\theta),(i,j)\notin X^*$, 满足$\lim\limits_{\theta\to\infty}\tau_{ij}^*(\theta)=0$.

依据上述命题, 由定理1, 2可以证明本文所提出算法的收敛性.

**定理 1**　设$\tau_{\min}>0,\mathrm{P}^*(\theta)$表示算法在前$\theta$次迭代中至少一次找到最优解的概率. 则对于一个任意少的数$\varepsilon>0$和一个充分大的$\theta$, 以下不等式成立: $\mathrm{P}^*(\theta)\geqslant 1-\varepsilon$, 并且$\lim\limits_{\theta\to\infty}\mathrm{P}^*(\theta)=1$. 即本文用到的蚁群算法只要运行到足够长的时间找到一个最优解的概率任意地接近1.

**定理 2**　令$\theta^*$为得到第1个最优解的迭代次数, 且令$\mathrm{P}(X^*,\theta,k)$为任意一只蚂蚁$k$在第$\theta$次迭代中得到最有解$X^*$的概率, 其中$\theta>\theta^*$. 那么以下等式成立: $\lim\limits_{\theta\to\infty}\mathrm{P}(X^*,\theta,k)=1$. 即每只给定的蚂蚁求出最优解的概率为1.

定理1和定理2的详细证明可参考文献[14]. 证明过程与文献[14]不同的是本文的路径构建方式不同, 因此证明过程不同.

## 4　实验结果分析(Analysis of experimental results)

在实验仿真中, 贴片机的贴片头个数$H$为4, 贴片头吸嘴之间的间距为16 mm, 供料槽之间的距离为16 mm, 两侧分别有50个共100个槽位(只考虑单边槽位). 表1为20块PCB板的参数, 其中主要包括每块PCB板上元器件种类和元器件的数目.

表 1 20块PCB板的参数
Table 1 Parameters of 20 PCBs

| 序号 | 种类 | 数目 | 序号 | 种类 | 数目 |
|---|---|---|---|---|---|
| 1 | 4 | 60 | 11 | 20 | 50 |
| 2 | 5 | 24 | 12 | 20 | 117 |
| 3 | 6 | 72 | 13 | 22 | 121 |
| 4 | 8 | 57 | 14 | 26 | 223 |
| 5 | 9 | 264 | 15 | 29 | 110 |
| 6 | 14 | 34 | 16 | 35 | 195 |
| 7 | 15 | 93 | 17 | 35 | 378 |
| 8 | 15 | 208 | 18 | 36 | 276 |
| 9 | 16 | 189 | 19 | 39 | 171 |
| 10 | 19 | 49 | 20 | 43 | 162 |

### 4.1 实验参数研究(Study of experimental parameters)

由于蚁群算法各参数紧密耦合, 参数的设定尚无严格的理论依据, 至今还没有确定最优参数的一般方法. 为了分析蚁群算法中各参数对算法搜索性能的影响, 随机选取了PCB5, PCB7, PCB9和PCB18 4块PCB板进行了测试, 实验采用改变一个参数, 其他参数不变的策略来讨论参数设置对算法性能的影响. 默认参数为: $\mathrm{Pant}=100$, $Q_1=Q_3=Q_4=10$, $Q_2=100$; 初始信息素: $\tau_1=\tau_2=0.1$, $\rho=0.5$, $\alpha=1$, $\beta=2$, $q_0=0.9$; 混合蛙跳的算法参数参数蛙群个数$Q=\mathrm{Pant}=100$; 模因组数$m=10$, $i_{\mathrm{part}}=5$, $i_{\mathrm{global}}=10$. 表2列出了第5块、第7块、第9块和第18块PCB板的蚁群算法参数配置的优化过程与结果. 表2中平均值和最优解分别为某个参数变化时算法运行10次的获得最短路径的平均值和最小值. 配置参数时, 参数的选取尽可能使平均值和最优解都能达到最小. 由表2可知, 在其他参数取默认值时, $\alpha=5$时所得的平均值和最优解比$\alpha$取其他值要好些; 同样$\beta=2$, $\rho=0.5$, $q_0=0.9$时所得的平均值和最优解比各参数取其他的值要好些, 因此对于PCB5参数最优配置为: $\alpha=0.5$, $\beta=2$, $\rho=0.5$, $q_0=0.9$. 同样可得PCB7参数的最优配置为: $\alpha=1$, $\beta=2$, $\rho=0.7$, $q_0=0.9$; PCB9参数的最优配置为: $\alpha=2$, $\beta=1$, $\rho=0.5$, $q_0=0.9$; PCB18参数的最优配置为$\alpha=0.1$, $\beta=1$, $\rho=0.5$, $q_0=0.9$.

混合蛙跳算法的参数比较简单, 随着蛙群个体数量增加, 算法能够得到更优解, 且模因组数对算法的影响小于蛙群个体数量[10]. 而在本文中随着蚂蚁数量的确定, 蛙群个数是固定不变的, 其他参数是参考文献[10]得到.

表 2 4块PCB板的参数配置与结果
Table 2 Parameter configuring and results of 4 PCBs

| 参数 | 参数值 | PCB5/mm | | PCB7/mm | | PCB9/mm | | PCB18/mm | |
|---|---|---|---|---|---|---|---|---|---|
| | | 平均值 | 最优解 | 平均值 | 最优解 | 平均值 | 最优解 | 平均值 | 最优解 |
| $\alpha$ | 0 | 48754.2 | 48603 | 14797.6 | 14695 | 37506.5 | 37253 | 59292.1 | 58582 |
| | 0.5 | 47794.3 | 47366 | 14485.4 | 14321 | 36767.3 | 36421 | 58177.1 | 56954 |
| | 1 | 47895.8 | 47607 | 14517.8 | 14174 | 36784.7 | 36313 | 57787.8 | 56997 |
| | 2 | 48147.2 | 47942 | 14488.5 | 14183 | 36535.6 | 36199 | 57658.3 | 56770 |
| $\beta$ | 0 | 52751.8 | 52346 | 15133 | 14781 | 37364.5 | 36874 | 60759.4 | 59516 |
| | 1 | 48134.9 | 47732 | 14445 | 14149 | 36428.8 | 36117 | 57307 | 56725 |
| | 2 | 47895.8 | 47607 | 14517.8 | 14174 | 36784.7 | 36313 | 57787.8 | 56997 |
| | 5 | 4815.2 | 47804 | 14564.6 | 14443 | 36951.1 | 136494 | 58082.6 | 57317 |
| $\rho$ | 0.3 | 47868 | 47582 | 14552 | 14346 | 36823.2 | 36474 | 57835 | 57337 |
| | 0.5 | 47895.8 | 47607 | 14517.8 | 14174 | 36784.7 | 36313 | 57787.8 | 56997 |
| | 0.7 | 47894.5 | 47677 | 14494.6 | 14303 | 36724.6 | 36353 | 57817.4 | 56447.1 |
| $q_0$ | 0.9 | 47895.8 | 47607 | 14517.8 | 14174 | 36784.7 | 36313 | 57787.8 | 56997 |
| | 0.7 | 47939.2 | 47910.4 | 14436.3 | 14305 | 36869.8 | 36335 | 57966 | 56916 |
| | 0.5 | 47830.3 | 47595 | 14515.9 | 14380 | 36968.9 | 36469 | 57938 | 57235 |

### 4.2 对比实验研究(Study of comparison experiment)

为验证本文设计的算法的有效性及运算效率, 在喂料器的位置确定且相同、每块PCB板的元器件种类和元器件数目相同情况下, 分别对表1中20块PCB板用混合蛙跳算法、蚁群算法和蚁群–混合蛙跳算法来优化元器件的贴片顺序. 由于每块PCB板的有不同的特性, 蚁群算法各参数的最优配置各有不同, 在进行算法的比较时, 蚁群混合蛙跳算法中参数采用默认的参数配置. 混合蛙跳算法中的参数采用文献[10]中的参数, 蛙群个数为360, 模因组数15, 局部迭代次数为10, 混合迭

代次数为100. 单一的蚁群算法中的参数采用蚁群–混合蛙跳算法的蚁群算法参数.

由于每次运行的结果不同, 针对每一块PCB板, 给出运行10次的统计结果, 并且对20块PCB的数据进行实验仿真, 结果如表3. 第1列为PCB板的序号, 第2列$F_{\rm s}$是为混合蛙跳算法的结果. 第3列$F_{\rm a}$为蚁群算法的结果, 第4列$F_{\rm as}$为本文算法结果. 第5列提高效率1通过式$\dfrac{F_{\rm s}-F_{\rm as}}{F_{\rm s}}\times 100\%$计算得到, 为混合蛙跳算法和本文算法比较提高的效率. 第6列提高效率2通过式$\dfrac{F_{\rm a}-F_{\rm as}}{F_{\rm a}}\times 100\%$计算得到, 为蚂蚁算法和本文算法比较提高的效率. 表3结果表明, 从算法整体性能看, 与文献[10]提出的改进的混合蛙跳算法相比, 本文算法得到平均解的效率提高了7.89%, 最优解的效率提高了6.30%, 与单一的蚁群算法相比, 本文算法得到的平均解的效率提高了3.79%, 最优解的效率提高了6.30%.

表 3 算法比较结果

Table 3 Algorithm comparison results

| 序号 | $F_s$/mm | | $F_{\rm a}$/mm | | $F_{\rm as}$/mm | | 提高效率1/% | | 提高效率2/% | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 平均 | 最优 | 平均 | 最优 | 平均 | 最优 | 平均 | 最优 | 平均 | 最优 |
| 1 | 10193.2 | 10040 | 9820.7 | 9752.4 | 9746.8 | 9675.1 | 4.38 | 3.63 | 0.75 | 0.79 |
| 2 | 3462.0 | 3392.8 | 3420.9 | 3380.6 | 3303.8 | 3271.8 | 4.57 | 3.57 | 3.42 | 3.22 |
| 3 | 12464.4 | 12119 | 12118 | 12004 | 11834.1 | 11691 | 5.06 | 3.53 | 2.34 | 2.61 |
| 4 | 11611.3 | 11461 | 11316.6 | 11266 | 11161.1 | 11057 | 3.88 | 3.53 | 1.37 | 1.86 |
| 5 | 48156.5 | 48000 | 48079.1 | 47874 | 47895.8 | 47607 | 11.99 | 11.00 | 0.38 | 0.56 |
| 6 | 6152.0 | 5933.7 | 5908.9 | 5845.3 | 5778.7 | 5700.5 | 6.07 | 3.93 | 2.21 | 2.48 |
| 7 | 16009.1 | 15416 | 14876.5 | 14696 | 14517.8 | 14174 | 9.32 | 8.06 | 2.41 | 3.55 |
| 8 | 43140.3 | 41606 | 40310.5 | 39422 | 39107.2 | 38666 | 9.35 | 7.07 | 2.99 | 1.92 |
| 9 | 39292.2 | 38246 | 38066.9 | 37595 | 36784.7 | 36313 | 6.38 | 5.05 | 3.37 | 3.41 |
| 10 | 8884.3 | 8591.6 | 9090.5 | 8713.0 | 8272.9 | 8070.3 | 6.88 | 7.38 | 8.99 | 7.38 |
| 11 | 9545.6 | 9406.8 | 9678.2 | 9487.9 | 9049.9 | 8935.4 | 5.19 | 5.01 | 6.49 | 5.82 |
| 12 | 23477.4 | 22597 | 20876.1 | 20851.1 | 20353.1 | 20172 | 13.31 | 10.73 | 2.51 | 3.26 |
| 13 | 21884.9 | 21025 | 20957.3 | 20484 | 20033.9 | 19549 | 9.24 | 7.55 | 4.41 | 4.57 |
| 14 | 24463.2 | 23390 | 22529.1 | 21858 | 21491.3 | 21163 | 12.15 | 9.52 | 4.61 | 3.18 |
| 15 | 39672.7 | 37969 | 38672 | 38028 | 36590.9 | 36329 | 7.78 | 4.32 | 5.38 | 4.47 |
| 16 | 86382.3 | 84111 | 80951.8 | 80039 | 79354.9 | 78612 | 8.16 | 6.54 | 1.97 | 1.78 |
| 17 | 57988 | 55016 | 53766.9 | 52796 | 52583.8 | 5163.9 | 9.32 | 6.19 | 2.20 | 2.19 |
| 18 | 65190.3 | 62860 | 59227.9 | 58065 | 57787.8 | 56997 | 11.36 | 9.33 | 2.43 | 1.84 |
| 19 | 42221.5 | 40955 | 4361.32 | 42722 | 3954.38 | 38933 | 6.34 | 4.94 | 9.33 | 8.87 |
| 20 | 36827.2 | 35655 | 37303.1 | 36212 | 34212.7 | 33851 | 7.10 | 5.06 | 8.29 | 6.52 |
| 20块PCB板平均效率 | | | | | | | 7.89 | 6.30 | 3.79 | 3.51 |

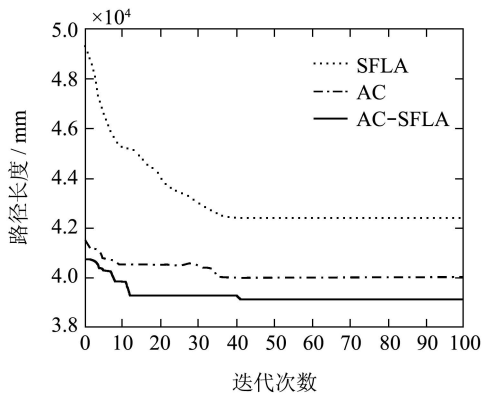


图 3 第8块PCB板的3种算法的搜索过程

Fig. 3 Search processes of three algorithms of No.8 PCB

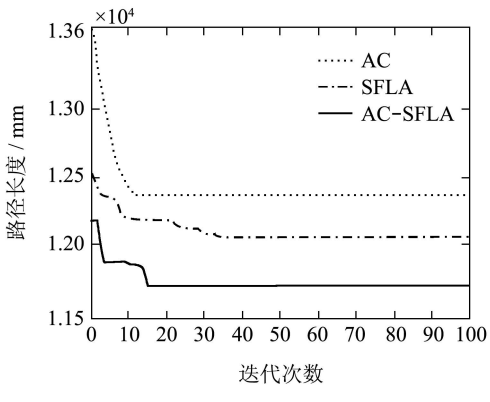


图 4 第3块PCB板的3种算法的搜索过程

Fig. 4 Search processes of three algorithms of No.3 PCB

图3和图4分别为表1中第8、第3块PCB板的3种算法的一次搜索过程和结果. 3种算法的搜索过程中初次迭代结果混合蛙跳算法(SFLA)比蚁群算法(AC)和蚁群–混合蛙跳算法(AC–SFLA)的初次迭代获得的最优结果要大, 这是由于在混合蛙跳算法中是青蛙个体是随机产生的造成的, 蚁群算方法和蚁群–混合蛙跳算法初次迭代结果依次变小. 图3中混合蛙跳算法求出的最优解的路径长度为42383 mm, 算法获得最优解得迭代数为40; 蚁群算法求出的最优解的路径长度为39950 mm, 算法获得最优解的迭代数为35; 而本文提出的蚁群–混合蛙跳算法中得到的最优解的路径长度为39116 mm, 算法获得最优解的迭代次数为41次. 因此表明, 蚁群–混合蛙跳算法比改进的混合蛙跳算法和单一的蚁群算法具有更高的求解精度和更强的全局搜索能力.

## 5 结语(Conclusions)

本文以多贴装头拱架式贴片机为研究对象, 针对喂料器的位置确定研究拱架式贴片机的元器件贴装顺序优化问题. 采用蚁群–混合蛙跳算法对贴片机路径优化进行求解. 在蚁群算法中, 针对贴片机实际贴装情况对启发函数和信息素进行分段, 以及采用了信息素的分段更新策略. 在蚂蚁搜索结果的基础上, 利用混合蛙跳算法的局部深度搜索和全局信息交换, 从而有效克服了蚁群算法容易陷入局部最优的弱点. 以20块实际生产的PCB为实例进行了测试, 实验表明蚁群–混合蛙跳算法比改进的混合蛙跳算法和单一的蚁群算法具有更高的求解精度和更强的全局搜索能力.

## 参考文献(References):


[1] LEE S H, LEE B H, T H PARK. A hierarchical method to improve theproductivity of a multi-head surface mounting machine[C] //*Proceedings of the 1999 IEEE International Conference on Robotics Automation*. New York: IEEE, 1999: 2110 – 2115.

[2] OR I, DEMIRKO L E. Optimization issues inautomated production of printed circuit boards: cperations sequencing andfeederconfiguration problems[C] //*The 1st IEEE International Conference on Emerging Technologies and Factory Automation*. New York: IEEE, 1995: 479 – 487.

[3] LIU H M, HU Y M. A heuristic optimization algorithm for multi-head mounter[C] //*Proceedings of the 22nd IEEE International Symposium on Intelligent Control Part of IEEE Multiconference on Systems and Control*. New York: IEEE, 2007: 279 – 384.

[4] WILLIAM H O, PING J I. A hybrid genetic algorithm for componentsequencing and feeder arrangement[J]. *Intelligent Manufacturing*, 2004, 15(3): 307 – 315.

[5] LI S Y, HU C F, TIAN F H. Enhancing optimal feeder assignment of the multi-head surface mounting[J]. *Applied Soft Computing*, 2008, 8(1): 522 – 529.

[6] LIN W Q, ZHU G Y. A genetic optimization approach to optimize the multihead surface mount placement machine[C] //*International Conference on Intelligent Robotics and Applications, Lecture Notes in Artificial Intelligence*-II. Berlin: Springer, 2008: 1003 – 1012.

[7] WILLIAM H O, PING J I. A genetic algorithm approach to optimising component placement andretrieval sequence for chip shooter machines[J]. *International Journal of Advanced Manufacturing Technology*, 2006, 28(5/6): 556 – 560.

[8] 张坤, 姜建国, 刘斌峰, 等. 贴片机路径优化研究[J]. 微计算机信息, 2009, 25(3/5): 196 – 198.
(ZHANG Kun, JIANG Jianguo, LIU Binfeng, et al. Research of route optimization for surface mount[J]. *Microcomputer Information*, 2009, 25(3/5): 196 – 198.)

[9] CHEN Y M, LIN C T. A particle swarm optimization approach to optimize component placement in printed circuit board assembly[J]. *International Journal of Advanced Manufacturing Technology*, 2007, 35(5/6): 610 – 620.

[10] 朱光宇. 模因内三角概率选择混合蛙跳算法[J]. 计算机集成制造系统, 2009, 15(10): 1979 – 1985.
(ZHU Guangyu. Meme triangular probability distribution shuffled frog-leaping algorithm[J]. *Computer Integrated Manufacturing Systems*, 2009, 15(10): 1979 – 1985.)

[11] 朱光宇, 林蔚清. 基于改进混合蛙跳算法的贴片机贴装顺序优化[J]. 中国工程机械学报, 2008, 6(4): 428 – 432.
(ZHU Guangyu, LIN Weiqing. Mounting sequential optimization on surface mounting machine using improved hybrid frog-jumping algorithm[J]. *Chinese Journal of Construction Machiner*, 2008, 6(4): 428 – 432.)

[12] 李英海, 周建中, 杨俊杰, 等. 一种基于阈值选择策略的改进的混合蛙跳算法[J]. 计算机工程与应用, 2007, 43(35): 19 – 21.
(LI Yingai, ZHOU Jianzhong, YANG Junjie, et al. Modified shuffled frog leaping slgorithm based on threshold selectionstrategy[J]. *Computer Engineer ing and Applications*, 2007, 43(35): 19 – 21.)

[13] 段海滨. 蚁群算法原理及应用[M]. 北京: 科学出版社, 2005.
(DUAN Haibin. *Principle and Application of Ant Colony Algorithm*[M]. Beijing: Science Press, 2005.)

[14] DORIGO Marco, STUTZLE Thomal. *Ant Colony Optimization*[M]. London: The Mit Press, 2004.


## 作者简介:


**陈铁梅** (1973—), 女, 讲师, 博士研究生, 从事智能优化研究, E-mail: mei57726@tom.com;

**罗家祥** (1979—), 女, 博士, 副教授, 主要从事智能优化研究;

**胡跃明** (1960—), 男, 教授, 博士生导师, 华南理工大学精密电子制造装备教育部工程研究中心主任, 主要从事精密电子制造方面的研发工作.