求解排列问题的分布估计离散粒子群优化算法

周雅兰[1],王甲海[2],黄 聪[2]
(1.广东财经大学信息学院,广东广州 510320;2.中山大学计算机科学系,广东广州 510006)

摘 要: 目前粒子群优化算法和分布估计算法较少用于解决排列编码组合优化问题,本文提出了一种新的适用于求解排列问题的分布估计离散粒子群优化算法.提出的算法结合粒子群优化算法和分布估计算法的思想,突破了标准粒子群优化算法速度-位移更新模式.新算法中每个粒子的信息一部分来自该粒子当前解排列与全局最优排列的最长公共子串,另一部分来自描述所有个体最优值分布信息的概率模型.这样粒子的当前解、所有个体最优值和全局最优值都参与了新解的生成过程,提出的算法秉承了粒子群优化算法的思想,同时具有更全面的学习能力,提高了算法的寻优能力以及避免陷入局部最优的能力.在两个经典的排列问题上的实验结果表明提出的算法具有良好的性能.

关键词: 离散粒子群优化;分布估计算法;排列问题
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2014)03-0561-11
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2014.03.021

Estimation of Distribution-Discrete Particle Swarm Optimization Algorithm for Permutation-Based Problems

ZHOU Ya-lan[1], WANG Jia-hai[2], HUANG Cong[2]
(1. *Information Science School*, *Guangdong University of Finance & Economics*, *Guangzhou*, *Guangdong* 510320, *China*;
2. *Department of Computer Science*, *Sun Yat-sen University*, *Guangzhou*, *Guangdong* 510006, *China*

Abstract: Particle swarm optimization algorithm (PSO) and estimation of distribution algorithm (EDA) are seldom applied to permutation-based combinatorial optimization problems. This paper presents an estimation of distribution-discrete particle swarm optimization algorithm (ED-DPSO) for the permutation-based problems. In ED-DPSO, one part of components of the offspring comes from the longest common subsequence between the current solution and the global best solution, and the other part comes from the probability model built on the distribution information of all personal best solutions. In ED-DPSO, the current solution, all personal best solutions and global best solution contribute to the generation of a new solution. Thus, ED-PSO has more comprehensive learning ability, and can avoid falling into local minima and improve the search ability. Experiment results on two classic permutation-based problems show ED-PSO has superior performance.

Key words: discrete particle swarm optimization; estimation of distribution algorithm; permutation-based problems

1 引言

粒子群优化算法(Particle Swarm Optimization,PSO)适用于解决连续优化问题,因流程简单和收敛速度快等特点受到了学术界的广泛关注[1~4].近几年,将 PSO 扩展应用到离散问题已经成为进化计算领域的研究热点之一,国内外学者先后提出了很多离散 PSO(Discrete PSO,DPSO),大致可分为两类:一类是保持标准 PSO 速度-位移的更新模式,通过修改公式各符号的意义与运算规则,使之适合求解离散问题,如最早的二进制版 PSO[5],量子离散 PSO[6];采用映射的方法[7];针对具体问题重新定义相关符号运算的方法[8,9];和基于集合的离散 PSO(Set-based DPSO,S-DPSO)[10].另一类是从 PSO 的更新原理出发,着重表达粒子的学习思想和过程,这类方法突破了速度-位移的更新模式,近几年来得到较大的发展,出现了一些较优秀的算法,比如利用遗传算法的思想[11,12];引入分布估计算法的思想[13,14].但是目前这些离散 PSO 的研究成果大多只针对解决二进制编码问题,较少用于排列编码问题.

分布估计算法(Estimation of Distribution Algorithms,EDA)[15]是一种新兴的基于概率模型的进化算法,近年来得到了广泛的研究与发展,并迅速成为解决实际工

收稿日期:2012-09-27;修回日期:2013-02-25;责任编辑:覃怀银
基金项目:国家自然科学基金(No.60905038,No.60805026,No.61070076);广东市珠江科技新星专项(No.2011J2200093,No.2012J2200085);广东商学院科研创新团队建设计划

程应用问题的有效方法.最初的 EDA 是基于二进制编码的离散算法,将 EDA 用于排列问题的研究成果较少[16,17].

近年,为了利用粒子群优化算法和分布估计算法的优点,提出了几种两者的混合算法,如用于解决连续优化的基于分布估计粒子群算法[18],用于解决 0-1 优化的基于分布估计粒子群算法[13,14].但迄今很少有用于解决排列编码问题的分布估计粒子群混合算法,其原因是由于前面所述的粒子群算法和分布估计算法分别在连续优化和 0-1 优化方面研究较多,而两者在排列问题上的应用研究较少.为了弥补和丰富这方面的研究,本文结合 PSO 和 EDA 的原理,提出一种适用于求解排列问题的分布估计离散粒子群优化算法(Estimation of Distribution-DPSO, ED-DPSO).提出的算法突破了标准 PSO 速度-位移的更新模式,粒子信息部分从当前解排列和全局最优值排列的最长公共子串中直接继承,剩余部分从描述所有个体最优值分布信息的概率模型中抽样获得.最长公共子串包含当前解和全局最优值的信息,增强了优秀基因对粒子的引导作用;概率模型包含所有个体最优值的信息,增强了粒子的全局信息学习能力,这使得算法具有防止过早陷入局部最优值的能力,提高了算法的寻优性能.实验以两个经典的排列组合优化问题-最小化总完工时间的流水作业调度问题和旅行商问题为例进行了实验,实验结果显示了提出算法解决排列问题时的良好性能.

2 标准粒子群优化算法

标准粒子群模型表示如下[2]:一个由 N 个粒子组成的种群,在 n 维空间搜索最优解,第 t 次迭代时粒子 i 的速度和位置分别表示为 $\boldsymbol{V}_i^t=(v_{i1}^t,\cdots,v_{id}^t,\cdots,v_{in}^t)$, $\boldsymbol{X}_i^t=(x_{i1}^t,\cdots,x_{id}^t,\cdots,x_{in}^t)$.每个粒子跟踪两个极值:个体最优值 $\boldsymbol{L}_i=(l_{i1},\cdots,l_{id},\cdots,l_{in})$ 和全局最优值 $\boldsymbol{G}=(g_1,\cdots,g_d,\cdots,g_n)$.第 t 代粒子 i 每一维状态更新的速度-位移公式如式(1)和式(2)所示:

$$v_{id}^{t+1}=\omega\cdot v_{id}^t+c_1\cdot r_1\cdot(l_{id}^t-x_{id}^t)+c_2\cdot r_2\cdot(g_d^t-x_{id}^t) \quad (1)$$

$$x_{id}^{t+1}=x_{id}^t+v_{id}^{t+1} \quad (2)$$

其中参数 ω 是惯性权值,表示粒子保持原来运动惯性的权重;c_1 和 c_2 是学习因子,分别表示粒子自身的经验(个体最优值)以及群体的经验(全局最优值)对粒子运动轨迹的影响;r_1 和 r_2 是[0,1]区间的随机浮点数.

3 分布估计算法

分布估计算法在遗传算法的基础上引入了统计学习的思想,通过对优秀个体进行信息统计,构建概率模型,然后抽样概率模型来生成新种群,取代了传统遗传算法的交叉、变异等操作,EDA 流程如图 1 所示[15].

```
随机生成初始种群;                                    //初始化
while(未达到终止条件){
  根据某选择策略从种群中选择若干个体;                  //选择
  根据这若干个体构建概率模型;                          //建模
  从概率模型中抽样生成新个体;                          //抽样
  新个体全部或部分替换原种群中的个体组成新种群.        //替换
  }
```

图 1 EDA 的一般流程

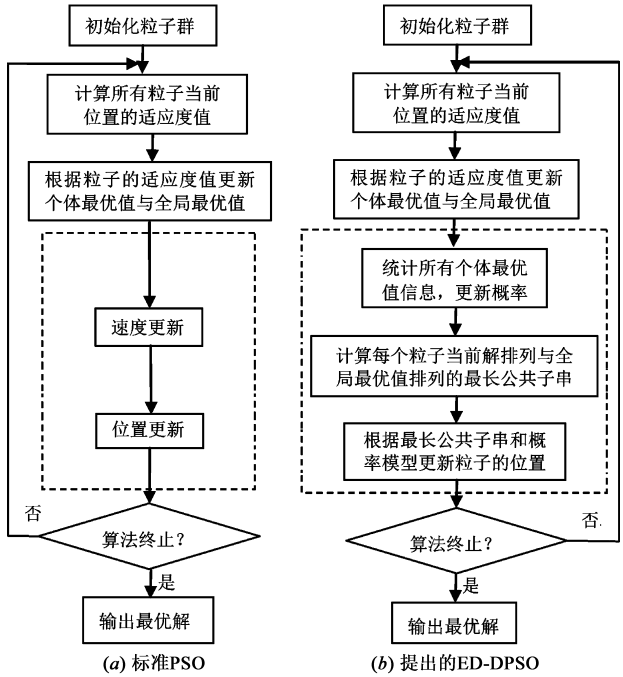
从算法流程可见,EDA 主要包括选择、建模、抽样和替换四步骤,其中最核心的是建立概率模型.算法依据概率模型及其更新引导种群进化,根据不同问题可以设计不同的概率模型及其更新机制.

4 分布估计离散粒子群优化算法

4.1 算法思想

PSO 的更新原理在于粒子可以不断地向自身的历史经验以及同伴的经验学习,获得更新信息从而调整自己的状态.从标准 PSO 的更新公式来看,算法的主要思想在于利用当前解、每个解的个体最优值以及全局最优值信息构造出新的解.EDA 的更新原理是根据有代表性的解在空间的分布情况进行统计学习来生成下一代解,用概率模型及其更新来实现统计学习.

结合 PSO 和 EDA 的更新原理,提出的算法根据所有粒子的个体最优值建立概率模型,即对种群中优质解的分布情况进行统计学习,概率模型的建立及其更新过程见 4.3 小节.提出的算法还引入粒子的位置排列与全局最优值排列的最长公共子串来引导算法进化,



(a) 标准PSO (b) 提出的ED-DPSO

图2 标准PSO与提出的ED-DPSO的流程比较

最长公共子串的求解过程见 4.4 小节.在状态更新过程中,粒子首先直接继承最长公共子串中的部分基因,然后抽样概率模型获得剩余基因,粒子状态更新过程见 4.5 小节.因此,提出的 ED-DPSO 秉承了 PSO 的思想,个体的当前解、所有个体的最优值和全局最优值都参与了个体状态的更新,大大增强了个体的全局信息学习能力,避免了过早陷入局部最优值的问题.

4.2　解的表示与种群初始化

ED-DPSO 用于求解排列组合优化问题,比如流水作业调度问题(Permutation Flow Shops Problem, PFSP)、旅行商问题(Traveling Salesman Problem, TSP)等等.因此,解简单地表示成一个排列,即 n 维问题的解可以表示为 $1,2,\cdots\cdots,n$ 的一个排列,这 n 个自然数的所有排列就构成了问题的解空间.

在 ED-DPSO 中初始种群采用随机的方式生成,尽可能地使初始解在搜索空间中分散,以保证种群的多样性.

4.3　概率模型及其更新

ED-DPSO 的概率模型 $\boldsymbol{P}$ 是一个 $n\times n$ 的矩阵,$\boldsymbol{P}=\begin{pmatrix}\boldsymbol{p}_{11} & \cdots & \boldsymbol{p}_{1n}\\ \vdots & \ddots & \vdots\\ \boldsymbol{p}_{n1} & \cdots & \boldsymbol{p}_{nn}\end{pmatrix}$,其中元素 p_{ij} 表示基因 i 在位置 j 的概率.初始时概率模型 $\boldsymbol{P}$ 中的所有元素 $p_{ij}=1/n$.采用 PBIL(Population-Based Incremental Learning)的更新方式[15].在粒子的状态更新之前,算法抽取所有粒子的个体最优值,构建出一个反映当前种群所有个体最优值空间分布的概率模型,概率模型 $\boldsymbol{P}$ 以学习速率 $\alpha\in[0,1]$ 向该模型进行学习,实现概率矩阵的平滑变动.概率模型 $\boldsymbol{P}$ 的更新公式如下:

$$p_{ij}=(1-\alpha)\cdot p_{ij}+\alpha\cdot\frac{1}{N}\cdot\sum_{k=1}^{N}\cdot I(l_{kj},i),1\leqslant i,j\leqslant n \tag{3}$$

$$I(l_{kj},i)=\begin{cases}1, & \text{if } l_{kj}=i\\ 0, & \text{otherwise}\end{cases} \tag{4}$$

其中 N 表示个体最优值的个数,即种群规模,l_{kj} 表示粒子 k 的个体最优值第 j 维分量的基因.式(3)和式(4)所描述的概率模型 $\boldsymbol{P}$ 中的每一个元素的更新,都需要遍历 N 个个体最优值的某一维基因,而 $\boldsymbol{P}$ 中总共有 $n\times n$ 个元素,因此更新 $\boldsymbol{P}$ 的时间复杂度为 $O(n^2\cdot N)$.

4.4　最长公共子串的求解

算法采用动态规划方法求解最长公共子串问题[19].假设有两个排列 $\boldsymbol{A}=(A_1,A_2,\cdots,A_p)$ 与 $\boldsymbol{B}=(B_1,B_2,\cdots,B_q)$,$C_{ij}$ 表示两个子排列 $(A_1,A_2,\cdots,A_i)$ 和 $(B_1,B_2,\cdots,B_j)$ 的最长公共子串,其中 $1\leqslant i\leqslant p,1\leqslant j\leqslant q$,那么存在如下关系:

如果 $A_i=B_j$,那么 $C_{i,j}=C_{i-1,j-1}\oplus A_i$.

如果 $A_i\neq B_j$,那么

$$C_{i,j}=\begin{cases}C_{i,j-1}, & \text{if } |C_{i,j-1}|>|C_{i-1,j}|\\ C_{i-1,j}, & \text{otherwise}\end{cases}.$$

其中,排列与元素的 $\oplus$ 运算表示该排列的末尾加上该元素组成一个新排列,比如 $(2,4,1,3)\oplus 5=(2,4,1,3,5)$,$C_{0,j}=C_{i,0}=\Phi$ 表示空串.

	B	1	4	2	7	3	5	6
A	0	0	0	0	0	0	0	0
2	0	0↑	1↑	1↖	1←	1←	1←	1←
4	0	0↑	1↑	1↑	1↑	1↑	1↑	1↑
6	0	0↑	1↑	1↑	1↑	1↑	↑	2↖
3	0	0↑	1↑	1↑	1↑	2↖	2↖	2↑
5	0	0↑	1↑	1↑	1↑	2↑	3↖	3←
7	0	0↑	1↑	1↑	2↖	2↑	3↑	3↑
1	0	1↖	1↑	1↑	2↑	2↑	3↑	3↑

图 3　最长公共子串生成示意图

假设粒子当前解排列 $\boldsymbol{A}=(2,4,6,3,5,7,1)$ 和全局最优值排列 $\boldsymbol{B}=(1,4,2,7,3,5,6)$,图 3 是 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的最长公共子串生成示意图.根据最长公共子串的求解流程[19],从最后一个格子开始,依据箭头指向往前推,即可得到粒子当前解排列 $\boldsymbol{A}$ 和全局最优值排列 $\boldsymbol{B}$ 的最长公共子串为(2,3,5).

4.5　粒子状态更新

在粒子状态更新过程中,粒子的新解首先继承最长公共子串中的基因,使得粒子可以保留自身较好的基因,但是最长公共子串同时也是全局最优值的子串,继承过多会使得粒子之间的相似度增加,破坏种群的多样性.为了保证种群的多样性,所有最长公共子串中的基因都以概率 $\lambda\in(0,1)$ 来继承,继承的基因直接从最长公共子串中复制到新解的相同位置上.粒子新解剩余的空位置从概率模型中抽样获得基因进行填充.最长公共子串的计算以及概率模型的更新在粒子状态更新过程之前完成,显然,单个粒子状态更新过程的时间复杂度为 $O(n^2)$.

第 $t+1$ 次迭代中粒子 $\boldsymbol{X}_k$ 的状态更新过程如下:

设 $I=(1,2,\cdots,n)$,$J=(1,2,\cdots,n)$,$\boldsymbol{X}_k^{t+1}$ 为空;

for $j=1:n$　　//遍历粒子当前值 $\boldsymbol{X}_k^t$ 的所有位置

　if(x_{kj}^t是最长公共子串中的基因 && rand()$\leqslant\lambda$)

　　直接复制 x_{kj}^t到 $\boldsymbol{X}_k^{t+1}$ 的位置j;

　　//继承最长公共子串中的基因

　　$I=I-\{x_{kj}^t\}$, $J=J-\{j\}$;

　end if

end for

while(J 不为空)

从 J 中随机选择一个位置j;

//随机选择一个空位置

以概率 $p_{ij}/\sum_{i\in I}p_{ij}$随机选择一个基因$i\in I$;

在 $\boldsymbol{X}_k^{t+1}$ 的位置j 插入基因 i; //从概率模型中抽样

$I=I-\{i\}$, $J=J-\{j\}$;

end while

return $\boldsymbol{X}_k^{t+1}$.

4.6 算法特点

标准 PSO 的所有粒子都从自身的个体最优值以及同一个全局最优值获得更新信息,所有的粒子都将追逐同一个最优个体,使得在算法中后期整个种群都局限在一个较小的空间范围内,容易陷入局部最优.

ED-DPSO 采用 EDA 抽取所有的个体最优值信息,构建了一个描述优质解空间分布信息的概率模型,粒子从概率模型的抽样中获得基因,也就是说粒子可以向所有的个体最优值信息进行学习,增强了粒子向全局信息学习的能力,避免了算法陷入局部最优.概率模型采用 PBIL 的更新方法,以一个学习速率向当前优质解的统计信息进行学习,即在算法的迭代过程中,所有粒子的个体最优值信息被统计出来更新概率模型,而算法又通过对更新的概率模型抽样来引导粒子状态的更新,所以,代表了粒子群优秀搜索经验(包括所有的个体最优值和全局最优值)的概率模型引导着整个种群的搜索与进化.

ED-DPSO 继承了当前解排列和全局最优值排列的最长公共子串中的基因,使得粒子可以直接继承自身的优秀基因,同时最长公共子串又是全局最优值的一个子串,因此加强了全局最优值在粒子群演化过程中的引导作用,提高了算法的寻优能力.

此外,ED-DPSO 参数较少,仅包含两个参数:概率模型的学习速率 α 以及最长公共子串的继承概率λ.学习速率 α 具有平衡算法全局勘探能力与局部开采能力的作用,α 越大局部开采能力越强,α 越小则全局勘探能力越强.参数 λ 具有平衡最长公共子串中基因继承力度与群体多样性的作用,λ 越大则继承较多最长公共子串中的基因,粒子之间的相似度也较高,破坏了种群的多样性,反之,种群的多样性得到了保证,但是粒子也将丢失大量的优秀基因,降低了算法的寻优能力.

5 仿真实验

为了测试算法的性能,ED-DPSO 在最小化总完工时间的 PFSP 和 TSP 两个经典的排列问题上进行了实验.实验中所有的算法都采用 C++ 语言编写,在 AMD Athlon 2.81 GHz CPU 以及 2GB 内存的台式机上运行.

下面首先简单介绍最小化总完工时间的 PFSP,接着对 ED-DPSO 的参数取值进行分析,然后 ED-DPSO 与最新最好求解该问题的 EDA 进行比较,再与离散 PSO 进行比较,通过这两方面的比较,来说明 ED-DPSO 相对于单一离散 EDA 和单一离散 PSO 的优势.最后还测试了 ED-DPSO 在 TSP 上的表现,进一步说明我们提出的算法可以很好的扩展至其它排列类组合优化问题,具有很好的通用性.

5.1 求解最小化总完工时间的 PFSP 的实验结果

5.1.1 最小化总完工时间的 PFSP 简介

PFSP 是一个经典的 NP 难问题,可以简单地描述为:有 n 个作业需要在 m 台机器上进行处理,对于一个排列,假设 $p_{i,j}$表示排列第j 个位置上的工作在机器i 上处理所需要的时间,$C_{i,j}$表示排列第 j 个位置上的工作在机器 i 上的完成时间,那么该排列的总完工时间 T_{sum} 的求解方法如下:

$$C_{i,0}=C_{0,j}=0 \tag{5}$$

$$C_{i,j}=\max\{C_{i,j-1},C_{i-1,j}\}+p_{i,j},1\leqslant i\leqslant m,1\leqslant j\leqslant n \tag{6}$$

$$T_{\text{sum}}=\sum_{j=1}^{n}C_{m,j} \tag{7}$$

最小化总完工时间的 PFSP 就是求 T_{sum}的最小值.

选用 Taillard 前 90 个问题实例进行实验,这 90 个实例按照规模大小($n\times m$)每 10 个实例一组共分成 9 组为:20×5(ta001 – ta010)、20×10(ta011 – ta020)、20×20(ta021 – ta030)、50×5(ta031 – ta040)、50×10(ta041 – ta050)、50×20(ta051 – ta060)、100×5(ta061 – ta070)、100×10(ta071 – ta080)、100×20(ta081 – ta090).

5.1.2 ED-DPSO 的参数分析

ED-DPSO 有两个参数:最长公共子串的继承概率 λ 以及概率模型的学习速率α.过大的 λ 值会破坏群体的多样性,过小的值又会使得粒子丢失大部分的优秀基因;α 值越大,算法的局部开发能力越强,值越小,算法的全局探索能力越强.

本文考查了 $\lambda=[0,0.2,0.4,0.5,0.6,0.8,1]$和 $\alpha=[0,0.01,0.02,0.05,0.08,0.1,0.5,1]$共 56 种参数组合下算法的表现.在这里为了简化表格,我们引入平均相对偏差值 $\Delta=\dfrac{V-S_{\text{best}}}{S_{\text{best}}}\times 100\%$ 来量化描述不同参数组合下 ED-DPSO 算法的表现,其中 V 表示算法对某一问题实例运行多次取得的平均值,S_{best}表示该实例迄今为止求得的最优值.可见,Δ 值越小,表示该算法的寻优能力越强.每种参数组合下的算法都对 90 个实例运行 10 遍得到 90 个 Δ 值,对这 90 个 Δ 值求平均值,得到表 1.算法的其他参数为:种群规模为 50,以生成 200000 个候选解作为算法的结束条件.从表 1 的数据可见,$\lambda=0.5$,$\alpha=0.02$ 时,算法取得的结果最好.

表 1　不同参数组合下 ED-DPSO 算法的表现

λ \ α	0	0.01	0.02	0.05	0.08	0.1	0.5	1
0	14.17	4.2718	3.73835	3.67302	3.69648	3.72655	3.93136	4.14104
0.2	14.0346	3.70206	3.33165	3.03196	2.95205	2.89095	3.1123	3.39531
0.4	13.739	3.15341	2.47303	2.46441	2.67297	2.73303	3.58352	4.01744
0.5	13.5758	2.67894	2.29999	2.67706	2.89771	2.98875	4.03837	4.51937
0.6	13.4166	2.33919	2.46965	2.91651	3.15599	3.2852	4.43955	4.94364
0.8	12.861	2.71559	3.04368	3.54844	3.85389	4.00648	5.10065	5.54812
1	11.2697	8.35351	7.80763	7.22725	7.07802	6.95791	7.00028	7.2126

参数 λ 具有平衡最长公共子串中基因的继承力度与群体多样性的作用.我们来考查 α 取值 0.02 时,不同的 λ 取值对群体多样性的影响.多样性采用文献[20]的定义:

$$\text{Diversity}(P_c)=\frac{\sum_{\pi_A,\pi_B\in P_c,\pi_A\neq\pi_B}\text{Hamming}(\pi_A,\pi_B)}{N\times(N-1)\times n} \tag{8}$$

其中,P_c 表示种群,N 表示种群的规模,n 表示排列的大小,并且有 $\text{Hamming}(\pi_A,\pi_B)=\sum_{i=1}^{n} f_n(\pi_{A,i},\pi_{B,i})$,以及 $f_n(\pi_{A,i},\pi_{B,i})=\begin{cases}0, & \pi_{A,i}=\pi_{B,i}\\ 1, & \text{otherwise}\end{cases}$.可见,$\text{Hamming}(\pi_A,\pi_B)$的最大值为 n,因此种群 P_c 的多样性的取值区间为[0,1],$\text{Diversity}(P_c)$值越大,种群 P_c 的多样性就越高.

算法对 90 个问题实例进行实验,每个实例运行 5 遍,每次运行的最大迭代数为 2000.记录每次迭代的群体多样性,并且对所有的实例求平均值,得到图 4.从图 4 可以看到,较大的 λ 值使得粒子之间具有较高的相似度,破坏群体多样性,较小的值带来较高的多样性.但是 λ 值越小,粒子丢失越多的优秀基因,降低了算法的寻优能力.从图中我们发现算法初始时的群体多样性都没有达到 1,只是近似于 1,那是因为种群的初始化是随机的,我们不能保证初始群体中任意两个个体上的任意位置上的基因都不相同.如图 5 所示,λ 过小与过大时算法的表现都很差,特别是取值为 1 时,粒子的基因全部继承最长公共子串,算法的表现最差.如图 6 所示,当 α 取值为 0 时,表示概率模型不具有学习的能力,永远保持初始时的平均值,由概率模型采样获得的解永远都是随机分散的,这时算法的表现最差.当 α 取值为 1 时,概率模型只反映当前种群的个体最优值信息,算法只具有局部开发能力,不具备全局探索能力,算法的表现也比较差.图 6 中有一种特殊的情况,即 α 取值为 0 时,λ 值越大,算法表现越好,当 λ 为 1 时,算法的表现最好,如图 7 所示.因为此时概率模型的采样等同于随机生成,算法只能通过粒子自身优秀基因的继承与积累而获得进化的能力.图 8 与图 9 是 ED-DPSO 在不同参数取值下的比较(对于每个问题实例,算法均运行 30 遍求得平均值).显然,当 λ = 0.5 以及 α = 0.02 时,算法具有较好的收敛性,参数过小或者过大都将导致算法过快收敛.

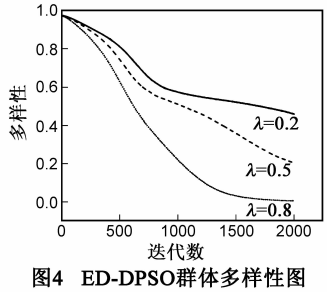


图4　ED-DPSO群体多样性图

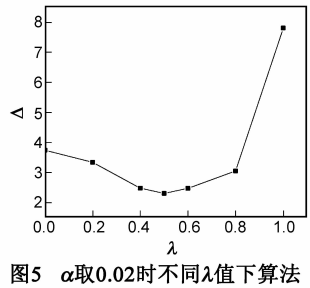


图5　α取0.02时不同λ值下算法性能

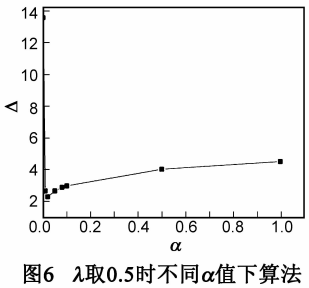


图6　λ取0.5时不同α值下算法性能

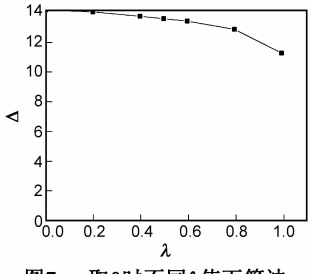


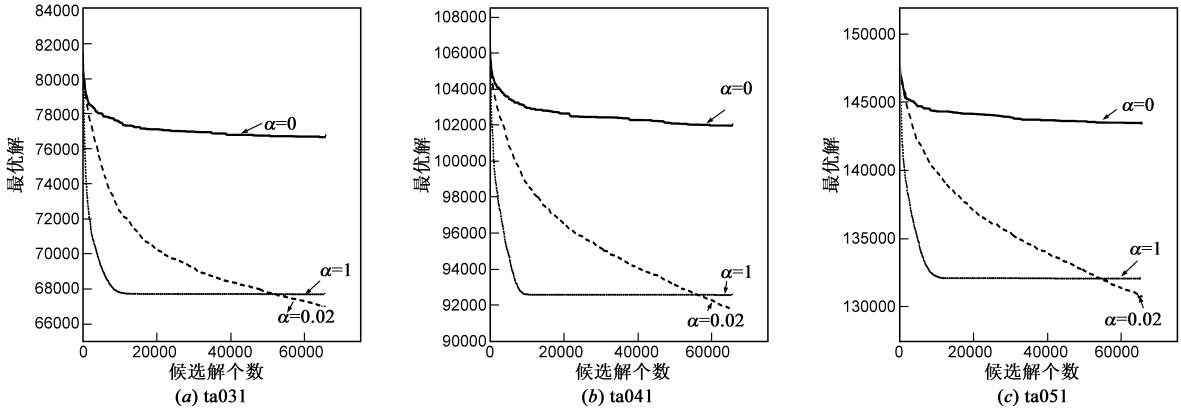
图7　α取0时不同λ值下算法性能

5.1.3　与求解排列问题的 EDA 实验比较

为了说明提出的算法相对于单一离散 EDA 的优势,ED-DPSO 与 ZEDA 进行了实验比较.ZEDA 是 Zhang 等人[20]提出的求解排列组合优化问题的 EDA,也采用

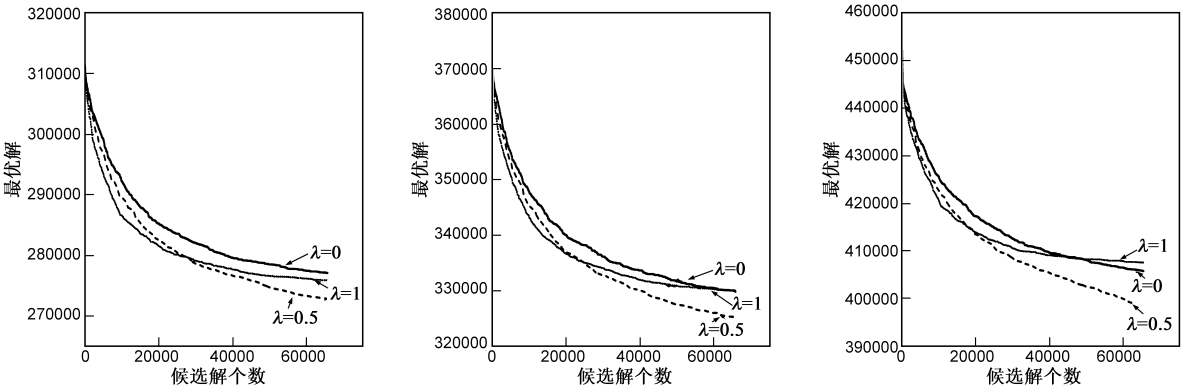
了当代个体解和种群最好解的最长公共子串信息生成下一代新个体解,是最近的性能最好的解决最小化总完工时间的 PFSP 算法.ZEDA 的参数设置[20]:$\alpha=0.3$,$\beta=0.2$,$\lambda=0.8$,以生成 200000 个候选解作为算法的结束条件.

表 2 是算法在每个实例上运行 50 次得到的最优值 Best、平均值 Mean 和标准差 Std..加粗字体表示两种算法中最好的结果.从表中可见在 90 个实例上 ED-DPSO 在最优值、平均值和标准差上均优于 ZEDA,这显示 ED-DPSO 具有比 ZEDA 更好的寻优能力和更稳定的性能.



(*a*) ta031 (*b*) ta041 (*c*) ta051

图8 不同α取值下的ED-DPSO算法的对比($\lambda=0.5$)



(*a*) ta061 (*b*) ta071 (*c*) ta081

图9 不同λ取值下的ED-DPSO算法的对比($\alpha=0.02$)

表 2 ZEDA 和 ED-DPSO 的比较

Instance		ZEDA	ED-DPSO	Instance		ZEDA	ED-DPSO	Instance		ZEDA	ED-DPSO
ta001	Best	14308	14034	ta031	Best	70409	65556	ta061	Best	281694	260127
	Mean	14712.4	14086.3		Mean	72800.3	65974.8		Mean	286832	261871
	Std.	223.76	42.236		Std.	1113.94	209.815		Std.	2803.48	1121.33
ta002	Best	15324	15151	ta032	Best	74947	69014	ta062	Best	272725	248623
	Mean	15868.2	15245.9		Mean	77130.2	69745.4		Mean	278107	250589
	Std.	214.183	44.6898		Std.	1394.27	252.18		Std.	2559.78	1181.61
ta003	Best	13769	13313	ta033	Best	68599	64279	ta063	Best	263002	242487
	Mean	14092	13491.3		Mean	71638.1	64710.9		Mean	270804	245223
	Std.	198.166	88.6197		Std.	1206.26	209.748		Std.	3377.86	1182.58
ta004	Best	15657	15447	ta034	Best	74213	69202	ta064	Best	254834	232723
	Mean	16226.3	15553.3		Mean	76261	69711.1		Mean	262527	234664
	Std.	267.946	57.4752		Std.	1026.78	241.835		Std.	3030.13	794.912

续表

Instance		ZEDA	ED-DPSO	Instance		ZEDA	ED-DPSO	Instance		ZEDA	ED-DPSO
ta005	Best	13669	13529	ta035	Best	75505	70131	ta065	Best	268577	246087
	Mean	14153.1	13601.5		Mean	77534.7	70641.5		Mean	274503	247582
	Std.	197.383	41.5515		Std.	995.481	223.626		Std.	3191.25	1088.97
ta006	Best	13507	13123	ta036	Best	72970	67692	ta066	Best	260252	238472
	Mean	14202.3	13227		Mean	75557.3	68182.1		Mean	267216	240412
	Std.	303.231	76.092		Std.	1046.19	269.477		Std.	3408.24	1328.56
ta007	Best	13890	13557	ta037	Best	72138	67141	ta067	Best	265526	246131
	Mean	14413.7	13693.1		Mean	74175.9	67659.5		Mean	273801	247801
	Std.	207.586	74.6664		Std.	914.498	281.522		Std.	3890.14	1170.38
ta008	Best	14356	13953	*ta*038	Best	69519	65398	ta068	Best	261230	237339
	Mean	14764.9	13999		Mean	73001.3	65738		Mean	268684	239547
	Std.	209.854	28.4183		Std.	1242.45	215.417		Std.	2842.92	1561.07
ta009	Best	14758	14315	ta039	Best	67721	63819	ta069	Best	278105	255052
	Mean	15300.3	14435.3		Mean	70203.4	64198.4		Mean	284003	256206
	Std.	275.336	88.7401		Std.	962.803	212.196		Std.	2740.72	744.543
ta010	Best	13197	12979	ta040	Best	74560	69733	ta070	Best	271168	249413
	Mean	13676.2	13076.3		Mean	76935.1	70236.2		Mean	279275	250754
	Std.	221.363	60.6384		Std.	1109.52	230.671		Std.	3443.32	879.882
ta011	Best	21572	20911	ta041	Best	95480	89045	ta071	Best	335860	307633
	Mean	22228.1	21095.7		Mean	98661.3	90189.6		Mean	344433	310979
	Std.	317.94	113.716		Std.	1257.48	554.526		Std.	3698.86	1208.95
ta012	Best	23181	22440	ta042	Best	90870	84884	ta072	Best	310486	283240
	Mean	23658.8	22840.6		Mean	93882.7	85666.2		Mean	318367	286821
	Std.	251.001	99.1026		Std.	1358.46	364.838		Std.	3769.33	1455.83
ta013	Best	20221	19872	ta043	Best	87756	81491	ta073	Best	324136	297571
	Mean	20891.6	19952.3		Mean	92410.4	82568.5		Mean	333964	300670
	Std.	315.139	51.0093		Std.	1619.28	645.007		Std.	3822.3	1290.22
ta014	Best	18995	18750	ta044	Best	93756	88324	ta074	Best	338277	310726
	Mean	19717	18951		Mean	97871.4	88978.3		Mean	345940	313963
	Std.	339.683	76.4927		Std.	1580.37	436.195		Std.	3198.13	1238.05
ta015	Best	19268	18713	ta045	Best	94656	88054	ta075	Best	322205	294599
	Mean	19848.2	18799.1		Mean	97991.8	89094.1		Mean	331054	296691
	Std.	338.236	60.1406		Std.	1506.29	482.163		Std.	2875.96	1012.86
ta016	Best	19543	19336	ta046	Best	94394	88150	ta076	Best	307843	279444
	Mean	20157.4	19463.3		Mean	97548.9	89095.6		Mean	315316	282410
	Std.	255.29	65.8523		Std.	1277.79	425.163		Std.	3850.44	1302.67
ta017	Best	18852	18376	ta047	Best	95944	90554	ta077	Best	317015	288021
	Mean	19420.9	18532.7		Mean	98644.9	91426.6		Mean	324784	290671
	Std.	323.089	87.6797		Std.	1281.46	381.327		Std.	3301.14	1143.17

续表

Instance		ZEDA	ED-DPSO	Instance		ZEDA	ED-DPSO	Instance		ZEDA	ED-DPSO
ta018	Best	20761	20268	ta048	Best	94847	88324	ta078	Best	326624	3000867
	Mean	21292	20456.1		Mean	97129	89419.6		Mean	333305	303125
	Std.	291.544	91.7053		Std.	980.687	343.493		Std.	3225.29	1334.01
ta019	Best	20616	20330	ta049	Best	94358	87391	ta079	Best	338081	310950
	Mean	21406.6	20438		Mean	96650.2	88317.3		Mean	344249	313617
	Std.	333.518	84.2237		Std.	1307.49	442.694		Std.	3553.82	1391.4
ta020	Best	21751	21357	ta050	Best	96719	90043	ta080	Best	331941	301056
	Mean	22247.1	21456.6		Mean	98869.9	90982		Mean	339135	304038
	Std.	259.815	59.5483		Std.	1177.9	387.121		Std.	3858.46	1376.83
ta021	Best	34394	33806	ta051	Best	136957	128412	ta081	Best	411060	377718
	Mean	35006.5	33985.2		Mean	139627	129766		Mean	419125	382074
	Std.	335.704	136.934		Std.	1195.53	652.087		Std.	4093.65	1527.34
ta022	Best	31956	31587	ta052	Best	130166	121914	ta082	Best	415136	384639
	Mean	32819.3	31783.6		Mean	133078	122981		Mean	422955	388907
	Std.	319.947	118.099		Std.	1327.57	566.947		Std.	4208.6	1643.28
ta023	Best	34496	33920	ta053	Best	128170	118702	ta083	Best	410138	382928
	Mean	35158.5	34150.8		Mean	131694	120152		Mean	421750	385364
	Std.	324.375	108.702		Std.	1511.98	731.022		Std.	4092.21	1375.18
ta024	Best	32065	31698	ta054	Best	30748	122814	ta084	Best	415633	386151
	Mean	32905.9	31814.5		Mean	133421	124203		Mean	423202	389475
	Std.	474.696	62.0223		Std.	1297.37	608.752		Std.	3541.45	1435.37
ta025	Best	34904	34616	ta055	Best	128975	120632	ta085	Best	412255	382067
	Mean	35632.1	34822.5		Mean	132651	121795		Mean	422105	385499
	Std.	366.879	93.4887		Std.	1557.6	580.05		Std.	4095.37	1702.54
ta026	Best	32858	32564	ta056	Best	131006	122918	ta086	Best	416260	384213
	Mean	33929.4	32756		Mean	133631	123986		Mean	424184	387688
	Std.	345.182	214.577		Std.	1299.49	457.045		Std.	4437.83	1525.57
ta027	Best	33209	33069	ta057	Best	133081	125155	ta087	Best	416141	385988
	Mean	34209.4	33243.9		Mean	135985	126513		Mean	426694	389440
	Std.	372.626	94.6077		Std.	1517.67	549.377		Std.	4469.95	1521.27
ta028	Best	32769	32456	ta058	Best	131039	124583	ta088	Best	423705	396637
	Mean	33606.5	32608.2		Mean	136011	126379		Mean	433495	399950
	Std.	331.611	97.2804		Std.	1724.91	651.696		Std.	3778.65	1565.21
ta029	Best	34158	33623	ta059	Best	131895	124160	ta089	Best	418122	387077
	Mean	34980.4	33847.4		Mean	134851	125195		Mean	424229	390885
	Std.	412.966	88.0936		Std.	1318.9	497.46		Std.	3212.9	1367.49
ta030	Best	32327	32269	ta060	Best	132682	126365	ta090	Best	424751	391957
	Mean	33484.9	32645		Mean	136794	127300		Mean	431336	395170
	Std.	418.593	141.907		Std.	1763.94	499.4		Std.	3396.3	1352.74

5.1.4 与求解排列问题的其他离散 PSO 实验比较

为了说明提出的算法相对于单一离散 PSO 的优势,ED-DPSO 与 S-DPSO 进行了实验比较. S-DPSO[10]将粒子的速度和位置都表示为集合,各项运算都统一为集合的运算,是最近的性能最好的求解排列问题的离散 PSO. 为了验证 ED-DPSO 引入当前解和全局最优值的最长公共子串信息能提高性能,ED-DPSO 与 GM-DPSO (Guided Mutation-DPSO)进行了对比实验. GM-DPSO 是 ED-DPSO 的一个变种,与 ED-DPSO 的不同之处是它没有引入最长公共子串机制,而是直接以概率 λ 复制全局最优值获得基因,即 Guided Mutation 操作[21]. 三种算法的参数设置如下:S-DPSO 的种群规模按文献[10]设为 20,ED-DPSO 和 GM-DPSO 设为 50;GM-DPSO 的 $\lambda = 0.2$, $\alpha = 0.02$;三种算法均以生成 200000 个候选解作为算法的结束条件,对每个实例都运行 50 遍. 在这里为了简化表格,我们只选择每个问题实例的平均相对偏差值 Δ 作为比较对象.

表 3 是对 90 个实例每组 10 个实例的平均相对偏差值 Δ 值求平均值的结果. 表中加黑字体表示三种算法中最好的结果. 从表中可见,GM-DPSO 与 ED-DPSO 都采用了 PBIL 的概率模型更新方式,较好地平衡了局部开采能力与全局勘探能力. 但是 GM-DPSO 的所有粒子都在同一个最优值附近的空间进行搜索,不利于维护种群的多样性,而 ED-DPSO 不仅采用了 PBIL 的概率模型更新方式,还采用了继承当前解和全局最优值的最长公共子串信息的方式,具有更优的寻优能力. S-DPSO 算法在问题规模较小时表现较好,但是规模较大时不如 ED-DPSO 和 GM-DPSO. 总体来说,ED-DPSO 的表现比其他两种离散 PSO 算法更好.

表 3　三种离散 PSO 的比较

Problem	S-DPSO	GM-DPSO	ED-DPSO
20×5	0.423473	1.4644	0.743224
20×10	0.683578	1.63976	0.944789
20×20	0.589039	1.25202	0.754288
50×5	4.04456	3.03301	2.0902
50×10	5.25593	4.08746	2.99392
50×20	4.96758	3.82462	2.83066
100×5	8.04669	3.40805	2.94675
100×10	9.23474	4.31132	3.65133
100×20	8.77234	4.21515	3.67387

5.2 求解 TSP 的实验结果

为了进一步验证算法的有效性,ED-DPSO 在另一个经典的排列组合优化问题-旅行商问题上进行实验,并与最近的求解该问题性能最好的 S-DPSO 算法[10]进行实验比较.

表 4　ED-DPSO 与 S-DPSO 的比较

Instance	Algorithm	ED-DPSO1	S-DPSO1	ED-DPSO2	S-DPSO2
eil51	439	Best	499	665	427
	Mean	573.1	704.88	438.8	448.74
	Std.	38.596	17.1157	5.39259	6.64172
eil76	Best	751	1180	545	779
	Mean	860.82	1240.46	574.18	844.38
	Std.	56.5571	30.302	13.5892	30.2601
kroA100	Best	41336	77695	23228	56438
	Mean	49341.4	84189.1	24881.8	61536.9
	Std.	4099.08	2345.16	923.499	1842.13
eil101	Best	1021	1782	672	1198
	Mean	1147.6	1882.3	700.02	1276.84
	Std.	73.698	38.3928	14.1485	32.9128
kroA150	Best	71465	146419	35056	109087
	Mean	81330	153744	41317.6	115046
	Std.	5247.8	2783.45	3172.04	2491.22
kroA200	Best	98503	215369	58353	171334
	Mean	113040	223866	64464.9	177452
	Std.	5336.89	2935.13	2215.71	2771.49

对于 TSP,S-DPSO 关于位置的元素(i,j)有两种表示形式:一是(i,j)表示第 i 个位置上的城市为j(S-DPSO1);二是(i,j)表示城市 i 与城市 j 的邻接边(S-DPSO2). 显然 S-DPSO1 算法具有更强的通用性,可适用于所有的排列组合优化问题,而 S-DPSO2 则引入了 TSP 的特点,具有针对性.

与两种 S-DPSO 相对应,对于 TSP,ED-DPSO 的概率模型元素 p_{ij}也有两种表示形式:一是 p_{ij}表示在位置j 上选择城市 i 进行填充的概率(ED-DPSO1);二是 p_{ij}表示城市 i 之后紧接着是城市 j 的概率(ED-DPSO2),这里 ED-DPSO2 算法统计城市邻接边的信息,最长公共子串表示粒子与全局最优值的公共邻接边.

四个算法的参数设置与前面的实验相同.

表 4 是四个算法的实验结果. 从表中可以看出,由于引入了与 TSP 相关邻接边的概念,ED-DPSO2 与 S-DPSO2 的表现分别比对应的 ED-DPSO1 与 S-DPSO1 要好. ED-DPSO1 比 S-DPSO1 具有更好的最优值 Best 和平均值 Mean,而 ED-DPSO2 的最优值 Best、平均值 Mean 和标准差 Std.(除 kroA150 实例外)是四个算法中最好的结果. 由此可见 ED-DPSO 具有比 S-DPSO 更好的寻优能力.

6 结论

提出适用于求解任意排列组合优化问题的 ED-DPSO,提出算法的新解部分来自粒子当前解排列与全局

最优排列的最长公共子串,剩余部分从描述所有个体最优值分布信息的概率模型中抽样获得,每个粒子具有更全面的学习能力,能避免算法陷入局部最优值从而提高寻优性能.在最小化总完工时间的流水作业调度问题上的实验结果表明,提出的算法具有比最新用于解决该问题的 EDA 和 DPSO 更优的性能.此外算法还在另一个经典排列问题-旅行商问题上进行了实验比较,表明算法具有较强的通用性.

参考文献

[1] KENNEDY J, EBERHART R C. Particle swarm optimization [A]. IEEE International Conference on Neural Networks[C]. Piscataway: IEEE Press, 1995. 1942 – 1948.

[2] SHI Y H, EBERHART R. A modified particle swarm optimizer [A]. IEEE World Congress on Computational Intelligence[C]. Piscataway: IEEE Press, 1998. 69 – 73.

[3] DEL VALLE Y, VENAYAGAMOORTHY G K, MOHAGHEGHI S, HERNANDEZ J C, HARLEY R G. Particle swarm optimization: basic concepts, variants and applications in power systems[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(2): 171 – 195.

[4] WU H, GENG J, JIN R, QIU J, LIU W, CHEN J, LIU S. An improved comprehensive learning particle swarm optimization and its application to the semiautomatic design of antennas[J]. IEEE Transactions on Antennas and Propagation, 2009, 57 (10): 3018 – 3028.

[5] KENNEDY J, EBERHART R C. A discrete binary version of the particle swarm algorithm[A]. IEEE International Conference on Systems, Man and Cybernetics[C]. Piscataway: IEEE Press, 1997. 4104 – 4108.

[6] YANG S Y, WANG M, JIAO L C. A quantum particle swarm optimization [A]. IEEE Congress Evolutionary Computation [C]. Piscataway: IEEE Press, 2004. 320 – 324.

[7] JARBOUI B, IBRAHIM S, SIARRY P, et al. A combinatorial particle swarm optimization for solving permutation flowshop problems [J]. Computers & Industrial Engineering, 2008, 54 (3): 526 – 538.

[8] 钟一文,蔡荣英.求解二次分配问题的离散粒子群优化算法[J].自动化学报,2007,33(8):871 – 874.
ZHONG Yi-wen, CAI Rong-ying. Discrete particle swarm optimization algorithm for QAP[J]. Acta Automatica Sinica, 2007, 33(8): 871 – 874. (in Chinese)

[9] 张长胜,孙吉贵,欧阳丹彤.一种自适应离散粒子群算法及其应用研究[J].电子学报,2009,37(2):299 – 304.
ZHANG Chang-sheng, SUN Ji-gui, OUYANG Dan-tong. A self-adaptive discrete particle swarm optimization algorithm [J]. Acta Electronica Sinica, 2009, 37(2): 299 – 304. (in Chinese)

[10] CHEN W N, ZHANG J, CHUNG H S H, et al. A novel set-based particle swarm optimization method for discrete optimization problems [J]. IEEE Transactions on Evolutionary Computation, 2010, 14(2): 278 – 300.

[11] 黄岚,齐季,谭颖,杨滨.一种求解矩形排样问题的遗传-离散粒子群优化算法[J].电子学报,2012,40(6):1103 – 1107.
HUANG Lan, QI Ji, TAN Ying, YANG Bin. A genetic-discrete particle swarm optimization algorithm for rectangular packing[J]. Acta Electronica Sinica, 2012, 40(6): 1103 – 1107. (in Chinese)

[12] 高海兵,周驰,高亮.广义粒子群优化模型[J].计算机学报,2005,28(12):1980 – 1987.
GAO Hai-bing, ZHOU Chi, GAO Liang. General particle swarm optimization model[J]. Chinese Journal of Computers, 2005, 28(12): 1980-1987. (in Chinese)

[13] 周雅兰,王甲海,印鉴.一种基于分布估计的离散粒子群优化算法[J].电子学报,2008,36(6):1242 – 1248.
ZHOU Ya-lan, WANG Jia-hai, YIN Jian. A discrete particle swarm optimization algorithm based on estimation of distribution[J]. Acta Electronica Sinica, 2008, 36(6): 1242 – 1248. (in Chinese)

[14] WANG J H, CAI Y Q, ZHOU Y L, et al. Discrete particle swarm optimization based on estimation of distribution for terminal assignment problems[J]. Computers & Industrial Engineering, 2011, 60(4): 566 – 575.

[15] BALUJA S. Population-based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning[R]. Pittsburgh: School of Computer Science, Carnegie Mellon University, 1994.

[16] 王圣尧,王凌,方晨,许烨.分布估计算法研究进展[J].控制与决策,2012,27(7):961 – 966.
WANG Sheng-yao, WANG Ling, FANG Chen, XU Ye. Advances in estimation of distribution algorithms[J]. Control and Decision, 2012, 27(7): 961 – 966. (in Chinese)

[17] CEBERIO J, IRUROZKI E, MENDIBURU A, LOZANO J A. A review on estimation of distribution algorithms in permutation-based combinatorial optim-ization problem [J]. Progress in Artificial Intelligence, 2012, 1(1): 103 – 117.

[18] EI-ABD M, KAMEL M S. A cooperative particle swarm optimizer with migration of heterogeneous probabilistic models [J]. Swarm Intelligence, 2010, 4: 57 – 89.

[19] CORMEN T H, LEISERSON C E, RIVEST R L, STEIN C. Introduction to Algorithms (2nd edition) [M]. Cambridge, MA: MIT Press, 2006.

[20] ZHANG Y, LI X P. Estimation of distribution algorithm for permutation flow shops with total flowtime minimization[J]. Computers & Operations Research, 2011, 60(4): 706 – 718.

[21] SALHI A, RODRíGUEZ J A V, ZHANG Q. An estimation of

distribution algorithm with guided mutation for a complex flow shop scheduling problem[A]. The 9th Annual Conference on Genetic and Evolutionary Computation(GECCO)[C]. New York:ACM Press,2010.570 – 576.

作者简介

周雅兰　女,1979 年 3 月出生于湖南省常德市.现为广东财经大学信息学院副教授.主要研究方向为人工智能与数据挖掘.
Email:zhouylan@163.com

王甲海　男,1977 年 6 月出生于江西省赣州市.现为中山大学计算机科学系副教授,从事人工智能及其应用的研究工作.
Email:wangjiah@mail.sysu.edu.cn